UNIVERSIDAD DE LA REPÚBLICA URUGUAY

FACULTAD DE INGENIERÍA

Plataforma basada en Gemelos Digitales para Dosificación de Precisión

Informe de Proyecto de Grado presentado por

María Belén Carozo Jorcin, Ana Belén Suárez Pungitore

en cumplimiento parcial de los requerimientos para la graduación de la carrera de Ingeniería en Computación de Facultad de Ingeniería de la Universidad de la República

Supervisores

Laura González
Silvana Pidre

Montevideo, 17 de diciembre de 2024

Agradecimientos

Con la presentación de esta tesis de grado, concluimos nuestra formación en la carrera de Ingeniería en Computación, de la Facultad de Ingeniería de la UDELAR, Montevideo, Uruguay. Este proyecto es el resultado de un gran esfuerzo, no solo de las que lo desarrollamos, sino también de muchas personas que realizaron aportes significativos desde sus distintos roles y lugares. Por eso queremos agradecer profundamente a nuestros familiares y amigos por el apoyo constante a lo largo de este proyecto y durante toda nuestra carrera. Estamos muy agradecidas con Manuel Ibarra y nuestras tutoras Laura González y Silvana Pidre, que colaboraron con esta tesis de grado y nos brindaron apoyo constante y aportes muy valiosos. Además nos gustaría agradecer a los integrantes del proyecto de grado Finglix 2.0: Darwin Araujo, Ignacio Daniel Vigna y Juan Ignacio Betarte, que desarrollaron su proyecto en simultáneo al nuestro y nos brindaron ayuda para integrar nuestro sistema con el suyo.

Por último, queremos reconocer la buena comunicación, disposición, y el apoyo mutuo que nos brindamos entre nosotras en este proyecto y durante toda la carrera. Sin duda fueron piezas clave que nos permitieron alcanzar esta meta juntas.

Resumen

La dosificación de precisión informada por modelos puede definirse como la individualización de los regímenes de tratamiento farmacológico basados en características del paciente que alteran la absorción del fármaco y respuesta del organismo. Tradicionalmente, para fármacos de estrecho margen terapéutico la dosis se ajusta en función de la exposición a nivel individual. Pero la exposición se determina únicamente con una medida de concentración en sangre, y se compara el valor con un rango terapéutico poblacional.

Los modelos farmacocinéticos permiten sacar más provecho de los datos individuales de los pacientes. Actualmente, existen varias herramientas que brindan soporte para la dosificación de precisión informada por modelos, pero generalmente presentan limitaciones en cuanto a integración con otras herramientas, así como en cuanto a extensibilidad y portabilidad de los modelos.

En Uruguay, se han creado varios modelos para este fin, los cuales se desarrollan con algunas de estas herramientas y están orientados a especialistas en el área. Proyectos de grado previos han avanzado en distintos aspectos del desarrollo de una plataforma que se integra con algunas de estas herramientas y brinda, de forma consolidada, funcionalidades tanto para especialistas del área como para utilizar en la clínica.

Por otro lado, las plataformas basadas en Gemelos Digitales se están visualizando cada vez más como herramientas que brinden soporte a la medicina de precisión. Este proyecto busca seguir avanzando en plataformas para la dosificación de precisión, en particular, aprovechando soluciones de Gemelos Digitales.

Se comienza realizando un análisis del uso de Gemelos Digitales en el área de la salud. En base a esto, se propone una solución que permita la dosificación de precisón informada por modelos pero centrada en la extensibilidad y portabilidad de los modelos y en el soporte de la misma mediante Gemelos Digitales. Se implementa una solución como prototipo utilizando Eclipse Ditto como soporte para Gemelos Digitales y Health Connect como intermediario entre *wearables* y la plataforma. El prototipo permite validar la factibilidad técnica y, con base en instancias de validación realizadas con potenciales usuarios, se comprobó la utilidad de plataformas de este tipo en la medicina actual.

Palabras clave: Dosificación, Simulación, Modelo farmacométrico, Gemelos Digitales, Wearables

Índice general

Glosario

Backend Backend Refieren a la capa de acceso a datos de una pieza de software, la infraestructura física o el hardware.

Background Estado en segundo plano de una aplicación móvil.

Broker Es un intermediario que facilita la comunicación entre diferentes sistemas.

Covariable Es una característica que es específica de un determinado individuo y que puede influir a la farmacocinética de un fármaco.

Docker Es una plataforma que permite crear, empaquetar y ejecutar aplicaciones en contenedores, entornos livianos que aseguran consistencia entre desarrollo y producción.

Frontend Refiere a la capa de presentación de una pieza de software, la infraestructura física o el hardware.

Login Se le llama al proceso de autenticación de los usuarios.

Wearable Refiere a cualquier dispositivo electrónico inteligente que se viste o se lleva en el cuerpo que se utilice para registrar y monitorear medidas de salud.

Siglas

API Application Programming Interface, interfaz de programación de aplicaciones.

CRA Create React App, herramienta para crear aplicaciones React.

CRUD Create, Read, Update, Delete. Crear, leer, actualizar y eliminar.

HTML HyperText Markup Language, lenguaje de marcas de hipertexto.

IoT Internet of Things, internet de las cosas.

LINS Laboratorio de Integración de Sistemas.

MIPD Model Informed Precision Dosing, dosificación de precisión informada por modelos.

NPM Node Package Manager, gestor de paquetes de Node.

OTP One Time Password, contraseña de un solo uso.

REST Representational State Transfer, transferencia de estado representacional.

Capítulo 1

Introducción

En este capítulo se brinda el contexto y la motivación del proyecto PGDT (Plataforma basada en Gemelos Digitales para Dosificación de Precisión), una plataforma destinada a la aplicación de Gemelos Digitales en la dosificación de precisión de fármacos. Se describe la importancia de la dosificación de precisión en fármacos, enfatizando en la importancia para los que poseen márgenes terapéuticos estrechos. Asimismo, se mencionan los proyectos existentes relacionados y la utilidad e innovación de aplicar Gemelos Digitales en este proceso. Finalmente, se establecen los objetivos de este proyecto, sus resultados y principales aportes.

1.1. Contexto y motivación

En la actualidad, la medicina de precisión se ha convertido en un pilar fundamental en el campo de la salud. Este enfoque, que busca adaptar los tratamientos médicos a las características individuales de cada paciente, está ganando cada vez más relevancia por su potencial para mejorar la eficacia de los tratamientos y reducir los efectos secundarios indeseados. La medicina de precisión se basa en la premisa de que cada individuo es único en términos de su genética, estilo de vida y entorno. Por lo tanto, los tratamientos médicos, especialmente la dosificación de medicamentos, deben ajustarse a estas particularidades individuales para lograr los mejores resultados posibles.

Para lograr este nivel de personalización, se han desarrollado diversos modelos y herramientas que ayudan a los profesionales de la salud a tomar decisiones más informadas. Estos modelos utilizan una variedad de datos, desde información genética hasta historial médico y factores ambientales, para predecir la respuesta de un paciente a un tratamiento específico. Sin embargo, muchos de estos modelos tienen limitaciones, ya sea en términos de la cantidad de datos que pueden procesar, la frecuencia con la que se actualizan, o la facilidad de uso para los médicos en su práctica diaria.

En este contexto, la tecnología de Gemelos Digitales emerge como una solu-

ción prometedora. Un gemelo digital es una representación virtual de un objeto o sistema físico que se actualiza en tiempo real (Ahmadi-Assalemi y cols., 2020). En el ámbito de la salud, un gemelo digital de un paciente podría incluir una amplia gama de datos, desde sus signos vitales hasta su historial médico completo. La importancia de mantener actualizada la información del paciente no puede subestimarse. Con un gemelo digital, los médicos podrían tener acceso a una visión completa y actualizada de la salud de sus pacientes en todo momento. Esto no solo mejoraría la toma de decisiones clínicas, sino que también permitiría una monitorización más efectiva de los tratamientos en curso. Además, la aplicación de Gemelos Digitales en la medicina de precisión (Kluwe y cols., 2021) ofrece la ventaja adicional de proporcionar datos reales y actualizados para las simulaciones de dosificación. Esto podría mejorar significativamente la precisión y la relevancia de estas simulaciones.

Actualmente, existen herramientas que ayudan a los químicos a generar simulaciones para dosis precisas de medicamentos (Drocco, Facal, y Pirotto, 2021). En particular, en el marco de un proyecto de grado de la Universidad de la República se desarrolla Finglix. Finglix ha demostrado ser muy útil en la investigación farmacéutica, ofreciendo avanzadas capacidades de simulación. Sin embargo, estas herramientas no resultan siempre amigables en la práctica médica diaria. Su enfoque tan especializado y su complejidad la hacen poco práctica para el uso rutinario por parte de los médicos. Además, no conecta directamente con los datos actualizados de los pacientes, lo que limita su capacidad para ofrecer recomendaciones de dosis personalizadas en tiempo real.

La motivación principal de este proyecto nace de la necesidad de superar las problemáticas de las herramientas actuales y aprovechar al máximo las nuevas tecnologías, p. ej Gemelos Digitales, para mejorar la práctica médica diaria.

Este proyecto busca llevar la medicina de precisión a otro nivel, ofreciéndoles a los médicos una herramienta completa que combine simulaciones avanzadas de dosificación con datos actualizados de los pacientes en tiempo real. Con esto, se busca mejorar la atención médica, optimizar los tratamientos y, como objetivo principal, obtener mejores resultados para los pacientes. Se espera que esto resulte en mejores resultados tanto a nivel de mejores recuperaciones médicas como a nivel del tiempo invertido en los tratamientos y consultas.

1.2. Objetivos

El objetivo general de este proyecto es desarrollar una plataforma basada en gemelos digitales que brinde soporte a la dosificación de precisión, tomando como base lo desarrollado en proyectos de grado previos realizados en el marco de Grupo Interdisciplinario para la Dosificación de Precisión (GIDP, s.f.). En particular, se busca complementar lo ya desarrollado en base a las capacidades que proveen las soluciones de Gemelos Digitales.

Para cumplir con el objetivo general se plantean los siguientes objetivos específicos:

- **Investigar las aplicaciones de los Gemelos Digitales en el ámbito**

de la salud. Dado que el área de los Gemelos Digitales se encuentra en rápida evolución, explorar sus posibles aplicaciones en el sector médico representa una contribución significativa. Se apunta a recabar información sobre farmacología y dosificación de precisión, plataformas existentes e integración con *wearables*.

- **Analizar la problemática planteada.** Este objetivo refiere al estudio del problema para poder entender las necesidades de los usuarios finales y producir una solución que se adecúe a las mismas. Comprende también el relevamiento de requerimientos, tanto funcionales (orientados a usuarios finales) como no funcionales, referentes a interoperabilidad, portabilidad y licenciamiento.

- **Diseñar una plataforma extensible basada en Gemelos Digitales.** Si bien este proyecto aborda necesidades específicas dentro del ámbito de la dosificación de precisión, el diseño de la plataforma debe contemplar una arquitectura flexible que permita su adaptación a otros contextos en el área médica. La extensibilidad se considera un aspecto central del diseño, asegurando que la plataforma pueda evolucionar y aplicarse a diferentes casos de uso en el futuro.

- **Implementar una plataforma para la dosificación de precisión basada en Gemelos Digitales.** Este objetivo refiere a la creación de un prototipo funcional basado en el diseño propuesto que resuelva la problemática planteada y cumpla con los requerimientos establecidos.

- **Realizar pruebas sobre la solución propuesta y validarla.** Este objetivo refiere a la evaluación de la plataforma desarrollada mediante pruebas unitarias, pruebas de integración, evaluaciones de código, además contando con una instancia con usuarios con el fin de validar la solución.

1.3. Aportes y resultados del proyecto

Los principales aportes y resultados del proyecto son:

- **Investigación de la aplicación del concepto de Gemelos Digitales al área de la salud.** Se analizan las aplicaciones prácticas del concepto de Gemelos Digitales en el ámbito de la salud. Este enfoque, relativamente nuevo, está ganando tracción en el campo de la informática y tiene el potencial de abrir numerosas puertas, por lo que este estudio busca contribuir a la investigación de las posibilidades. Se documenta la información y el análisis relacionados con el uso de Gemelos Digitales en el ámbito de la salud, explorando sus beneficios, desafíos y potencial para mejorar el monitoreo y tratamiento de los pacientes.

- **Identificación de necesidades.** Se relevan y documentan los requerimientos de la plataforma, tanto funcionales como no funcionales, con el

fin de comprender en profundidad las necesidades de los centros de salud y los profesionales médicos. Este análisis apunta a contemplar que la plataforma responda adecuadamente a las demandas del entorno clínico.

- **Diseño de la solución**. Se propone una solución a la problemática planteada, detallando los componentes del sistema necesarios para definir una plataforma que se beneficie del uso del concepto de Gemelos Digitales para el área de la salud, y en particular el área de dosificación de precisión.
- **Implementación de plataforma basada en Gemelos Digitales para control de dosificación en pacientes y gestión de los mismos.** Se desarrolla y despliega en la nube una plataforma web extensible que cumple la función de proveer al personal de la salud una interfaz gráfica para gestionar tratamientos personalizados de pacientes, monitorear la información de salud del paciente en tiempo real y simular diferentes dosis usando estos datos para adecuar el tratamiento. Además, se desarrolla una aplicación móvil para gestionar el intercambio de datos entre los *wearables* y la plataforma.
- **Verificación y validación**. Se diseñan y ejecutan distintas pruebas sobre la plataforma implementada para corroborar la calidad y la usabilidad de la misma, que servirán de base para evitar regresiones y para ser extendidas a medida que la plataforma evoluciona. Además, se realizan instancias de intercambio con potenciales clientes para evaluar la solución planteada con respecto a las necesidades manifestadas y obtener comentarios sobre posibles mejoras e incorporaciones a futuro.
- **Contribución al Grupo Interdisciplinario en Dosificación de Precisión (GIDP).** El trabajo comprendido en este proyecto contribuye al GIDP y al área médica en general, acercando la Dosificación de Precisión a los profesionales de salud con una plataforma con mayor soporte gráfico que las existentes y generando soluciones que utilizan tecnologías de vanguardia para la medicina mediante el uso de Gemelos Digitales.

1.4. Organización del documento

El resto del documento se organiza de la manera detallada a continuación.

En el Capítulo 2 se presenta el marco conceptual que brinda información clave para el entendimiento del dominio y de las decisiones tomadas.

El Capítulo 3 describe el relevamiento y análisis de requerimientos del sistema, analizando las necesidades del proyecto y describiendo los requerimientos funcionales y no funcionales.

En el Capítulo 4 se aborda el diseño de la solución. Se describe la arquitectura general del proyecto, se presenta el modelo relacional utilizado y se detallan algunas pruebas de concepto realizadas.

El Capítulo 5 corresponde a la etapa de implementación. Se describen las etapas, herramientas y procesos involucrados en la construcción del *frontend*, el *backend*, la aplicación móvil y la configuración de los Gemelos Digitales.

En el Capítulo 6 se muestra el proceso de pruebas y validaciones realizadas. Se describen las pruebas efectuadas para corroborar el correcto funcionamiento del sistema y verificar la calidad del mismo, así como también se muestran los resultados de las validaciones con usuarios.

En el Capítulo 7 se especifican las etapas y la gestión del proyecto. Se detalla el tiempo aproximado y en qué orden se desarrollan las etapas, y se compara el cronograma original con el cronograma real.

Finalmente, en el Capítulo 8 se presentan las conclusiones obtenidas a partir del desarrollo del proyecto. Se resumen los resultados alcanzados, se evalúa el cumplimiento de los objetivos y se reflexiona sobre posibles mejoras y trabajos futuros.

Capítulo 2

Marco Conceptual

En este capítulo se exponen los conceptos clave necesarios para la comprensión del documento y que resultan relevantes para el desarrollo del proyecto.

2.1. Farmacología

Flower (2013) explica que la farmacología, clave en la medicina moderna, es la ciencia que estudia todo sobre los medicamentos y cómo interactúan con el cuerpo. Desde el origen de los fármacos hasta sus efectos, se investiga tanto sus propiedades físicas y químicas como sus impactos en el organismo. No solo se enfoca en cómo los medicamentos afectan al cuerpo, sino también en cómo el cuerpo los procesa, lo que ayuda a desarrollar terapias más efectivas y seguras.

Un punto clave en la farmacología es la dosificación, que consiste en definir la cantidad correcta de medicamento, cada cuánto tomarlo y por cuánto tiempo. Dar con la dosis adecuada es muy importante para que el tratamiento sea efectivo y, al mismo tiempo, minimizar los efectos secundarios. Esto es especialmente importante en la medicina personalizada, donde se trata de ajustar los tratamientos a cada paciente según sus características individuales, ajustando dosis y planes.

En resumen, la farmacología sigue siendo una disciplina en constante cambio y de suma importancia para que la medicina avance. Su papel en el desarrollo de tratamientos seguros y efectivos es clave, especialmente en esta nueva era de la medicina personalizada. Con la llegada de nuevas tecnologías como los Gemelos Digitales y la inteligencia artificial, este campo promete cambiar aún más, ofreciendo tratamientos más precisos y eficientes. El futuro de la farmacología presenta muchas oportunidades, donde la mezcla de la ciencia médica y las nuevas tecnologías tiene el potencial de abrir muchas puertas para mejorar la salud y el bienestar.

2.1.1. Modelos

El surgimiento de modelos para la simulación de medicamentos y sus efectos en el cuerpo ha cambiado por completo el mundo de la farmacología y el desarrollo de fármacos. Estos modelos, cada vez más avanzados, ayudan a investigadores y médicos a predecir mejor cómo van a interactuar los medicamentos con el cuerpo, lo que mejora el desarrollo de nuevos fármacos y las estrategias de tratamientos personalizados.

Como explican Kariya y Honma (2024), estos modelos se basan en complejos algoritmos matemáticos y computacionales que intentan imitar los procesos del cuerpo humano. Integran una gran cantidad de datos, desde las propiedades moleculares de los medicamentos hasta parámetros del paciente, para predecir la eficacia, toxicidad y cómo se comporta el fármaco en el cuerpo. El autor menciona varios tipos de modelados.

El modelado farmacocinético/farmacodinámico (PK/PD) es uno de los enfoques más usados hoy en día. Este modelado combina la farmacocinética, que se encarga de explicar cómo el cuerpo procesa el medicamento, con la farmacodinámica, que analiza los efectos del medicamento en el cuerpo. Los modelos PK/PD pueden predecir la concentración del fármaco en diferentes partes del cuerpo con el tiempo y relacionar esas concentraciones con los efectos que produce, lo que ayuda a ajustar las dosis de manera más óptima.

Otro enfoque importante es el modelado basado en la fisiología (PBPK). Estos modelos dividen el cuerpo en diferentes compartimentos que representan órganos y tejidos, y usan ecuaciones para describir cómo se mueve el medicamento entre ellos. Los modelos PBPK son muy útiles para predecir interacciones entre medicamentos y para aplicar datos de un grupo a otro, como de animales a humanos o de adultos a niños.

En el campo de la farmacogenómica, los modelos de simulación utilizan datos genéticos para predecir cómo las variaciones en el ADN de cada persona pueden influir en la forma en que responden a los medicamentos. Estos modelos son clave para avanzar hacia una medicina totalmente personalizada, ya que permiten ajustar las dosis y elegir los medicamentos más adecuados según el perfil genético del paciente.

Por otro lado, los modelos de dinámica molecular y *docking* molecular son muy importantes en las primeras etapas del descubrimiento de nuevos fármacos. Estos modelos simulan a nivel atómico cómo interactúan los medicamentos con sus objetivos moleculares, lo que ayuda a los investigadores a predecir la afinidad y eficacia de los compuestos antes de tener que sintetizarlos físicamente.

Los modelos de inteligencia artificial también están ayudando a crear «pacientes virtuales» más avanzados. Estos modelos *in silico* permiten simular cómo un paciente reaccionaría a diferentes tratamientos, lo que da a los médicos la oportunidad de probar varias opciones antes de aplicarlas en la realidad. Esta metodología tiene el potencial de reducir los riesgos relacionados con los ensayos clínicos y acelerar el desarrollo de nuevos medicamentos.

En el futuro, se espera el desarrollo de modelos más completos que integren datos a diferentes niveles, desde lo molecular hasta todo el cuerpo. Estos modelos

multi-escala podrían ofrecer una comprensión más profunda de cómo los medicamentos afectan al organismo, teniendo en cuenta las complejas interacciones entre los distintos sistemas biológicos.

2.1.2. Dosificación de Precisión Informada por Modelos

Como mencionan Kluwe y cols. (2021), la aprobación de medicamentos suele basarse en medidas tales como exposición, respuesta y variabilidad en poblaciones estudiadas. Estas medidas excluyen a pacientes con comorbilidades, pacientes que toman otros medicamentos o los que están muy enfermos; lo que puede llevar a una falta de representación de la diversidad de las poblaciones.

Esta falta de representación genera una alta variabilidad en los resultados terapéuticos individuales de cada paciente.

El enfoque de Dosificación de Precisión Informada por Modelos (MIPD por sus siglas en inglés), surge como una solución a este problema. Busca generar dosis personalizadas para las necesidades de cada paciente, usando un marco Bayesiano para calcular parámetros individuales del modelo. Estos se basan en conocimientos previos de la droga, que se individualizan mediante «covariables», como pueden ser: peso, sexo, edad, otra medicación, entre otras.

Si bien este enfoque mejora ampliamente los resultados si es comparado con métodos *a posteriori*, todavía no ha tenido una gran adopción por parte de la comunidad médica dado que es un área de investigación novedosa que se encuentra en pleno crecimiento.

Es posible que MIPD no sirva para todos los medicamentos: los caros, los que pueden tener efectos adversos severos o los que varían mucho entre personas, son los que se beneficiarían mayoritariamente del enfoque. Además, para obtener los mejores resultados, tiene que existir una proporcionalidad directa entre la concentración de la droga y el efecto clínico que esta produce.

La cantidad disponible de datos de salud de pacientes solo tiende a aumentar, gracias a *wearables*, biosensores y dispositivos de muestreo en casa. Considerando este volumen creciente de datos y los avances en métodos de análisis de datos que se puedan dar en el futuro para MIPD, se podría alcanzar una dosificación de precisión mejor informada y automatizada.

Kluwe y cols. (2021) mencionan también que el *software* amigable al usuario y la integración en el flujo de trabajo clínico son aspectos cruciales para terminar de derribar las barreras restantes y que se puedan desbloquear todos los beneficios de las herramientas MIPD.

El área médica se vería beneficiada de entender que se debe reemplazar la noción de "una dosis sirve para todos los pacientes" para mejorar el resultado de los tratamientos individuales. Además, las herramientas MIPD brindan apoyo y recomendaciones, pero la decisión final siempre sería tomada por el profesional.

La informática debe adaptarse también al enfoque MIPD, no solo para usar los modelos sino también para proporcionar información que los mejore. Sin embargo, la traducción de los resultados de investigaciones académicas a herramientas de *software* fáciles de usar es una parte desafiante del proceso de

adopción del enfoque. Esto puede ser particularmente difícil si los centros de salud tienen grandes cantidades de información en papel, mucha diversidad en las soluciones tecnológicas utilizadas, o pocos recursos destinados a la capacitación de personas para el uso de soluciones MIPD.

2.2. Fuentes de datos de salud

En esta sección se explican las diferentes fuentes de datos de salud que fueron consideradas para integrar al proyecto.

2.2.1. Wearables

Las tecnologías *wearables* en el área del cuidado de la salud incluyen los dispositivos electrónicos que el usuario puede usar, es decir, que son portátiles como relojes inteligentes, pulseras inteligentes y todo dispositivo que esté diseñado para recolectar información de la salud y del ejercicio del usuario. Éstos también pueden incluso enviar información a un médico o a otro profesional de la salud en tiempo real.

Como menciona Y. Wurmser (2023) la tecnología de *fitness* portátil es un área que ha ganado un espacio significativo en la industria de la salud ya que hay un interés creciente de los consumidores por monitorear su propia salud, lo que ha triplicado en los últimos cuatro años el uso de *wearables*. Esta demanda se espera que continúe aumentando aún más. El mercado de usuarios de dispositivos inteligentes *wearable* en los Estados Unidos se prevé que crecerá un 25.5 % interanual en 2023, frente al crecimiento del 23.3 % interanual en 2021, según un pronóstico de octubre de 2021 de Insider Intelligence que es el proveedor de referencia para pronósticos, datos y perspectivas en los campos de marketing, publicidad y comercio (*Insider Intelligence*, 2023).

Ejemplos de *wearables* más comunes según Y. Wurmser (2023):

- **Pulseras inteligentes.** Los rastreadores de *fitness* portátiles son pulseras equipadas con sensores para llevar un registro de la actividad física y la frecuencia cardíaca de un usuario. En el mercado actual son los *wearables* más simples pero proveen gran sincronización con aplicaciones de teléfonos inteligentes que a su vez llegan hasta a proveer recomendaciones de salud al usuario. Un ejemplo de este es el FitBit Flex (*Fit Bit Flex*, 2023) que en su última versión ofrece también una variedad de funciones de seguimiento de la salud para mejorar el sueño.

- **Relojes inteligentes.** Los relojes inteligentes antes eran utilizados únicamente para dar la hora y contar los pasos. Hoy en día, p. ej. Apple, ha lanzado un producto que, junto con una aplicación, monitorea el ritmo cardíaco de los usuarios y alerta a aquellos que experimentan fibrilación auricular. Además algunos de sus modelos tienen sensores de temperatura. Es decir, ofrecen lo que brinda una pulsera inteligente más algunas novedades.

- **Monitores de Electrocardiograma.** Los monitores de Electrocardiograma (ECG) portátiles tienen la capacidad de medir electrocardiogramas o ECGs, lo que los distingue de los relojes inteligentes. Business Insider (*Business Insider*, 2023) informó que Withings ganó el premio al mejor dispositivo portátil en la Feria de Electrónica de Consumo de 2019 con su producto Move ECG. El Move ECG puede medir un electrocardiograma y enviar la lectura al médico del usuario, además de detectar la fibrilación auricular. También puede realizar un seguimiento de la velocidad, la distancia y la elevación, así como el seguimiento automático de caminatas, carreras, natación y ciclismo.

- **Monitores de la presión arterial.** En 2019, Omron Healthcare lanzó HeartGuide, el primer monitor de presión arterial portátil. Aunque puede parecer un reloj inteligente típico, HeartGuide es un monitor de presión arterial oscilométrico que puede medir la presión arterial y la actividad diaria, como los pasos dados, la distancia recorrida y las calorías quemadas. HeartGuide puede almacenar hasta 100 lecturas en su memoria y todas las lecturas se pueden transferir a una aplicación móvil correspondiente llamada HeartAdvisor para su revisión, comparación y optimización del tratamiento. Los usuarios de HeartAdvisor tienen la capacidad de almacenar, hacer un seguimiento y compartir sus datos con su médico, al mismo tiempo que obtienen información para determinar cómo sus hábitos personales afectan su presión arterial.

- **Biosensores.** Los biosensores son dispositivos médicos portátiles emergentes que son radicalmente diferentes de los relojes inteligentes. Son especialmente útiles en medicina para el diagnóstico de enfermedades, el seguimiento de pacientes y la investigación biomédica debido a su capacidad para proporcionar mediciones precisas y específicas de sustancias biológicas en tiempo real. Se utilizan para detectar y medir la presencia o concentración de sustancias biológicas, como biomoléculas, células, microorganismos o cambios en las características fisiológicas, mediante la conversión de una señal biológica en una señal eléctrica o química que se puede medir y analizar.

  El biosensor portátil de Philips es un parche autoadhesivo que permite a los pacientes moverse mientras recopila datos sobre su movimiento, frecuencia cardíaca, frecuencia respiratoria y temperatura. La investigación realizada en el Augusta University Medical Center mostró que este dispositivo portátil registró una reducción del 89 % en la deterioración de los pacientes hacia paros cardíacos o respiratorios prevenibles. Esto demuestra la capacidad que tienen los dispositivos portátiles para mejorar los resultados de los pacientes y posiblemente reducir la carga de trabajo del personal médico.

2.2.1.1. Wearables Epidérmicos

Como se menciona en el artículo de Liu y cols. (2023), existe un tipo de *wearables* llamado epidérmicos los cuales se colocan directamente sobre la piel y se adhieren de manera flexible. Estos dispositivos están diseñados para monitorear diversas señales fisiológicas, como la temperatura corporal, el ritmo cardíaco, la actividad muscular, la hidratación, e incluso la concentración de ciertos biomarcadores en el sudor.

- **Fuentes de biofluidos.** Utilizadas para la detección epidérmica y su correlación con la concentración sanguínea de biomarcadores y medicamentos. Se consideran: sudor, lágrimas, saliva, entre otros. La correlación de concentraciones entre biofluidos y sangre es crítica. Se necesita más investigación para ampliar el monitoreo y establecer bases para detección con dispositivos portátiles.

- **Sensores ópticos.** Estos sensores utilizan cromóforos o fluoróforos para detectar cambios de color en biofluidos y realizar análisis cuantitativos. La detección colorimétrica, aunque útil, enfrenta limitaciones como la la falta de reactivos, lo que dificulta su aplicación en el monitoreo terapéutico continuo de medicamentos. Por otro lado, la detección de fluorescencia, que es más sensible que la colorimetría, enfrenta problemas como la fotodegradación, y requiere entornos oscuros para obtener resultados precisos.

2.2.1.2. Wearables Invasivos

A su vez, en el artículo de Liu y cols. (2023) se menciona otro tipo de *wearables*, llamados *wearables* invasivos. Estos dispositivos necesitan alguna clase de intervención quirúrgica o penetrar en el cuerpo para poder funcionar. Estos dispositivos se implantan en órganos, tejidos internos o bajo la piel y están diseñados para monitorear constantemente parámetros del cuerpo, administrar medicamentos o interactuar directamente con el organismo para mejorar la salud o el rendimiento.

2.2.1.3. Wearables para el monitoreo de drogas y enfermedades

La tecnología portátil permite monitorear en tiempo real cómo varían las concentraciones de medicamentos, lo que mejora la precisión en la detección de datos inusuales y riesgosos. En esta subsección se presentan las tecnologías para detectar moléculas de medicamentos y los sensores portátiles que se han desarrollado para hacer seguimiento de fármacos para tratar el Parkinson, antibióticos, analgésicos y neurolépticos, según se menciona en Liu y cols. (2023).

- **Drogas para tratar el Parkinson.** El texto menciona el uso de sensores portátiles para el monitoreo continuo de L-Dopa, un medicamento utilizado en el tratamiento de la enfermedad de Parkinson. La enfermedad afecta a las neuronas dopaminérgicas en el cerebro, causando síntomas motores.

Aunque la L-Dopa es efectiva, su farmacocinética puede variar según factores como la dieta y la edad. El exceso de L-Dopa puede tener efectos secundarios, y se destaca la importancia de sensores portátiles para un monitoreo preciso y ajustes de dosis individuales.

Se menciona la detección de L-Dopa a través del sudor como un enfoque prometedor. Se describen sensores portátiles basados en enzimas capaces de medir la L-Dopa en el sudor, lo que facilita un seguimiento continuo y no invasivo. Estos sensores destacan por su estabilidad (mantienen su rendimiento de manera confiable a lo largo del tiempo) y por la correlación de sus lecturas con las concentraciones de L-Dopa en la sangre. Esto significa que las mediciones en el sudor reflejan de manera precisa los niveles del medicamento en el cuerpo, lo cual es valioso para ajustar el tratamiento y mejorar la calidad de vida de los pacientes.

- **Drogas contra microbios.** Los autores abordan el papel vital de los antibióticos en el tratamiento de enfermedades infecciosas y la dificultad en determinar dosis adecuadas debido a las variaciones en la farmacocinética de los pacientes. Aunque hay tecnologías de detección *in vitro* para antibióticos, el monitoreo mediante algún *wearable* solo se ha implementado para algunos, como vancomicina, kanamicina, tobramicina y fenoximetilpenicilina. Para cada uno se han desarrollado sensores avanzados para detectarlos de manera precisa y no invasiva. Se destaca la importancia del monitoreo continuo para mejorar la precisión y la eficacia del tratamiento.

  El texto subraya la importancia de la detección portátil de antibióticos para optimizar dosis y mitigar la resistencia a los medicamentos, especialmente dada la prevalencia de bacterias resistentes a los antibióticos. Se destaca la promesa de las técnicas de detección no invasivas y portátiles para mejorar la eficacia terapéutica y reducir los riesgos asociados con la terapia combinada de antibióticos.

- **Drogas analgésicas.** El texto aborda el monitoreo de acetaminofén (APAP) y fentanilo mediante sensores portátiles. El APAP, propenso a la sobredosis, puede causar daño hepático irreversible. Sensores portátiles han sido desarrollados para detectar APAP en sudor y saliva mediante voltametría de pulso diferencial (DPV). También se ha creado un sensor integrado en guantes de plástico que detecta APAP en sudor y saliva, mostrando excelente flexibilidad y límites de detección clínicamente relevantes.

  En cuanto al fentanilo, un opioide sintético, se desarrolló un sensor electroquímico de microneedle portátil que analiza picos redox para detectar concentraciones de fentanilo. Estos avances en sensores portátiles que utilizan fluidos corporales como muestras de detección permiten generar curvas farmacocinéticas en tiempo real para medicamentos analgésicos, prometiendo aplicaciones futuras en monitoreo médico para intervenir y prevenir eventos adversos asociados con sobredosis de medicamentos.

- **Drogas psicoactivas.** El autor aborda la preocupación global por el abuso de drogas psicoactivas y la importancia del monitoreo de la concentración sanguínea de estas sustancias para garantizar la seguridad y eficacia del tratamiento. Se destaca el desarrollo de sensores portátiles que utilizan el sudor como muestra para detectar drogas psicoactivas, empleando tecnologías como la electroquímica o la espectroscopía Raman mejorada en superficie.

  Se menciona la detección de cafeína, la sustancia psicoactiva más consumida, utilizando un sustrato flexible con un electrodo modificado con nanotubos de carbono para medir los niveles en el sudor. También se aborda la detección de catinonas sintéticas, una clase de sustancias psicoactivas, mediante sensores electroquímicos basados en aptámeros.

  El texto resalta la importancia del monitoreo de drogas psicoactivas y la necesidad de desarrollar tecnologías de detección altamente selectivas para abordar la diversidad de estas sustancias y el uso concurrente de múltiples sustancias por parte de los usuarios. Se enfatiza la contribución potencial de estas tecnologías al trabajo de monitoreo de la salud.

2.2.2. Historia clínica electrónica en Uruguay

En Uruguay, la Agencia de Gobierno Electrónico y Sociedad de la Información y del Conocimiento (AGESIC) de la Presidencia de la República posee un programa llamado Salud Digital. Este programa tiene como cometido prinicipal incorporar el uso de las Tecnologías de la Información y la Comunicación (TIC) en el área de la Salud para «mejorar la calidad y continuidad asistencial de todas las personas que viven en el país» (Salud Digital, s.f.). Además, en la página oficial del Gobierno, se explica que Salud Digital se crea como una evolución del Programa Salud.uy, nacido en 2012 en cooperación con Presidencia de la República, Ministerio de Salud Pública (MSP), el Ministerio de Economía y Finanzas (MEF) y AGESIC.

Desde sus inicios, se describe en Salud Digital que el Programa Salud.uy reunió a personas del ecosistema de salud de Uruguay para trabajar en estrategias de informática médica, siempre con un enfoque en las personas. Para lograrlo, creó espacios de colaboración técnica y organizacional que ayudaron a formar y fortalecer una comunidad interdisciplinaria, que integra tanto al sector público como al privado y se alinea con las políticas nacionales de salud. Salud.uy estableció reglas y pautas en informática médica para que la Historia Clínica Electrónica Nacional (HCEN) sea posible y segura. Esto permite que los equipos de salud puedan acceder a la información de cada paciente en tiempo real y desde cualquier lugar del país. Así, la atención es más precisa y de mejor calidad.

La definición brindada por el Ministerio de Salud Pública en (Historia Clínica Electrónica Nacional, s.f.) es la siguiente: *"La Historia Clínica Electrónica Nacional (HCEN) es una plataforma que permite acceder a la Historia Clínica Digital de los usuarios del sistema de salud y que posibilita el registro de*

cualquier evento médico, independientemente del lugar geográfico y prestador del salud en donde se dé la asistencia. Permite el intercambio de información clínica con fines asistenciales entre prestadores de salud, con el fin de asegurar la continuidad asistencial del usuario en el Sistema Nacional Integrado de Salud (SNIS).”

Además, se menciona que los usuarios del SNIS mayores de 18 años pueden acceder a su Historia Clínica Digital y ver todos los eventos asistenciales registrados. También pueden compartir su información clínica con los equipos de salud que los atiendan, sin importar en qué parte del país estén o a qué prestador de salud pertenezcan. De acuerdo con la normativa vigente, el personal de salud puede acceder a estos registros digitales en dos situaciones: cuando el usuario está en una consulta o evento asistencial, o si se presenta una emergencia médica. Solo con permitir el acceso a información clínica centralizada de los distintos niveles de atención, ya se mejora la toma de decisiones durante el proceso de atención. Cuando los miembros del equipo de salud pueden ingresar directamente las indicaciones (en lugar de solo transcribirlas), se les da la oportunidad de interactuar justo en el momento en que los sistemas clínicos ofrecen información útil, lo que mejora la prescripción (ayudando a evitar duplicar estudios o ingresar dosis incorrectas, entre otras cosas). Además, este sistema ayuda a seguir mejor las guías de práctica clínica gracias a alarmas y recordatorios.

Además, se cuenta en (Friedmann, s.f.) que los principios que rigen la HCEN incluyen la finalidad, veracidad, completitud, reserva, información y accesibilidad. Su objetivo principal es mejorar la continuidad de la atención y la calidad del cuidado de la salud. Para lograr esto, se establecieron dos objetivos estratégicos: todos los prestadores deben tener una historia clínica electrónica que garantice la interoperabilidad, y la HCEN debe crear un entorno de confianza que permita un acceso seguro a la información clínica cuando sea necesario. Para hacer todo esto posible, se creó una red privada de alta velocidad llamada Red Salud, que conecta a todos los prestadores de salud. Esta red se diseñó tomando como modelo la Red.uy, que conecta a organismos estatales y ofrece normas de seguridad y un único punto de acceso a los servicios estatales. También se reutilizaron algunos componentes, como la plataforma de interoperabilidad del gobierno y los mecanismos de autenticación, para conectar tanto a prestadores públicos como privados. La Red Salud tiene sus propias políticas de seguridad y un monitoreo activo, garantizando que los datos se intercambien en un entorno privado con altos estándares de seguridad y confidencialidad

En Uruguay, según se menciona en la página oficial de AGESIC (AGESIC, s.f.), la HCEN está regulada por varias leyes y normativas que garantizan su implementación y funcionamiento, así como la protección de datos personales y la confidencialidad de la información. Algunas de las leyes y regulaciones relevantes son las siguientes:

- **Ley N.º 18.331 de Protección de Datos Personales.** Esta ley, promulgada en 2008, establece las condiciones para el tratamiento de datos personales, incluyendo los datos de salud. La ley protege los derechos de los titulares de datos y regula cómo las entidades pueden recolectar, al-

macenar y procesar información personal.

- **Ley N.º 19.660 de Historia Clínica Electrónica.** Esta ley, aprobada en 2018, establece el marco legal para la implementación de la HCEN en el país. Define las características, objetivos y requisitos para el manejo de la historia clínica electrónica, promoviendo su uso en el sistema de salud uruguayo y asegurando la interoperabilidad entre diferentes instituciones de salud.
- **Ley N.º 19.167 de Derechos y Deberes de los Pacientes.** Esta ley, que protege los derechos de los pacientes, también aborda aspectos relacionados con la historia clínica y la confidencialidad de la información médica. Establece que los pacientes tienen derecho a acceder a su historia clínica y a que se respete la privacidad de su información.
- **Reglamento de la Ley N.º 19.660.** Este reglamento complementa la ley de HCEN, proporcionando detalles sobre la gestión y el manejo de la información en el contexto de la historia clínica electrónica, incluyendo aspectos técnicos y operativos.
- **Normativas del Ministerio de Salud Pública (MSP).** El MSP ha emitido diversas resoluciones y guías para la implementación y uso de la HCEN, estableciendo estándares y procedimientos para garantizar la seguridad y confidencialidad de los datos de salud.

Estas leyes y regulaciones forman un marco legal que busca promover el uso de la historia clínica electrónica en Uruguay, garantizando la protección de los datos personales y los derechos de los pacientes. Se observa además, la importancia del sector médico en el país ya que es un tema abordado por varias instituciones y en continúo progreso tanto a nivel legal como a nivel tecnológico.

2.3. Gemelos Digitales

Según Ahmadi-Assalemi y cols. (2020), el concepto de Gemelos Digitales vincula artefactos físicos individuales con modelos digitales que reflejan su estado en tiempo real. Lehner y cols. (2022) lo definen como una representación virtual de un sistema físico que facilita la comunicación bidireccional entre el sistema real y su modelo digital.

En esta sección se discuten los beneficios de los Gemelos Digitales para el caso de uso de este proyecto, así como también un análisis de posibles herramientas a usar para la implementación de los mismos.

2.3.1. Aplicación en la salud

Tradicionalmente, el área de la salud y cuidado médico ha tenido un enfoque reactivo. Es decir, en lugar de centrarse en la prevención, se actúa sobre los problemas de salud de los pacientes a medida que aparecen. Esta forma de

operar tiene un costo: no solo aumenta la posibilidad de que la gravedad de la condición empeore por ser tratada de forma tardía, sino que también implica un gasto económico para el centro de salud. (OMS, 2005)

En la actualidad, según Ahmadi-Assalemi y cols. (2020), esta característica de la medicina está gradualmente siendo sustituida por un enfoque más centrado en el bienestar general de los pacientes y en la prevención. Las nuevas líneas de investigación y soluciones informáticas contribuyen a esta evolución, pero los avances tecnológicos se suelen adoptar lentamente en la medicina.

Sin embargo, mencionan que es probable que las tecnologías digitales vayan abriéndose camino cada vez más en la salud. Monitorear los signos vitales y la información cambiante de los pacientes desde sus casas es mucho más eficiente que depender de consultas médicas poco frecuentes.

Como mencionan los autores, el concepto de Gemelos Digitales emerge para permitir el modelado y la fusión de artefactos físicos individuales (p. ej., *wearables*) con modelos digitales que reflejan su estado en tiempo real. El término «medicina de precisión» describe la idea de generar tratamientos especialmente pensados para cada individuo, y un gemelo digital de un paciente puede cumplir un rol muy importante para lograr alcanzar mayores niveles de personalización en dichos tratamientos, entender mejor los riesgos a los que se encuentra sometido el paciente y definir intervenciones.

2.3.1.1. Limitaciones y desafíos

Un gemelo digital puede utilizarse, p. ej., para monitorear y gestionar enfermedades crónicas. Se sabe que los pacientes con estas condiciones varían significativamente debido a diversos factores, en particular los bio-psico-sociales. Por lo tanto, una implementación ideal de Gemelos Digitales debe considerar estas variables, teniendo en cuenta que factores como las decisiones relacionadas con el estilo de vida no son fáciles de medir o monitorear. Además, no basta con la infraestructura tecnológica, tiene que existir una colaboración entre el médico y el paciente. Este intercambio no solo es necesario para corroborar que el programa está funcionando para el paciente, sino también para enriquecerlo con sus propias experiencias y hacerlo más aplicable a su propia vida.

Este modelo también podría implementarse de manera que envíe consejos y recordatorios basados en las medidas automáticas tomadas del paciente y los datos de comportamiento, con el objetivo de evitar complicaciones, mejorar la calidad de vida del paciente y reducir los costos para el sistema de salud.

En casos agudos, como accidentes graves, puede ser más difícil porque no solo es necesario basarse en guías clínicas, sino también considerar otros factores como la genética, la tolerancia, entre otros. Además, estos casos agudos pueden requerir decisiones urgentes.

2.3.1.2. Implicancias éticas

Al implementar productos médicos con Gemelos Digitales, se deben considerar factores relacionados con minorías sociales, estereotipos y otros sesgos

para evitar errores de diagnóstico. El gemelo digital debería tener en cuenta estos aspectos y evitar perpetuar estereotipos, asegurando un proceso libre de discriminación.

Con estas tecnologías se busca ayudar a la población y hacer de los cuidados médicos un beneficio más accesible, pero si no se tiene cuidado se puede lograr el efecto contrario y aumentar la brecha (Bruynseels, de Sio, y van den Hoven, 2018).

También es de suma importancia la ciberseguridad y el cuidado de la privacidad y confidencialidad de los datos de los pacientes. No sólo por tratarse de información de salud, sino también debido al uso de *wearables* y sensores implantados, que tienen un impacto directo en el bienestar del paciente. Para cumplir con esto de la mejor manera posible, es crucial tener un enfoque fuerte en ciberseguridad desde el inicio. También tiene que haber un compromiso de transparencia, consentimiento informado y privacidad para mantener la confianza del paciente.

Por otra parte, el proceso de recolección de datos puede ser bastante costoso o incluso perjudicial. p. ej., ¿debería repetirse un tratamiento aunque exista riesgo para el paciente con el fin de recolectar datos, o debería basarse solo en resultados ya reportados por el paciente?

2.3.1.3. Ejemplo real de caso de uso

Powers y cols. (2021) realizan un análisis de cómo los relojes inteligentes y los sistemas de monitoreo continuo pueden ser útiles en el seguimiento de enfermedades tales como el Parkinson.

En el artículo se señala que la calidad de vida de los pacientes con Parkinson está relacionada con la capacidad de los médicos para ajustar con precisión las medicaciones. Aunque las terapias de reemplazo de dopamina mejoran los síntomas motores, determinar los horarios óptimos de medicación sigue siendo un reto. Para medir la gravedad de los síntomas motores, se emplea la revisión de la parte III de la escala MDS-UPDRS, una herramienta ampliamente aceptada. Sin embargo, las evaluaciones actuales son limitadas, ya que se basan en visitas clínicas esporádicas y en la memoria de los pacientes, lo que puede llevar a errores.

En este contexto, una solución que combine medidas de relojes inteligentes con Gemelos Digitales podría ser muy beneficiosa para el tratamiento de los pacientes enfermos de Parkinson.

2.3.2. Implementación de Gemelos Digitales

La noción de Gemelos Digitales es un concepto, lo que significa que hay diversas opciones a la hora de implementarlo. Se puede optar por una implementación propia o utilizar un proveedor que brinde la infraestructura básica para soportar Gemelos Digitales. En esta sección se analiza el esfuerzo de llevar a cabo una implementación propia, se exploran las posibilidades y beneficios de

usar las distintas plataformas, y finalmente se decide qué opción se ajusta mejor a las necesidades del proyecto.

2.3.2.1. Implementación propia

Para tener una solución a medida de Gemelos Digitales, se puede recurrir a una implementación propia. El concepto de Gemelos Digitales es amplio, pero debería cumplir con una serie de requisitos básicos (Lehner y cols., 2022) para adaptarse a nuestras necesidades:

- **Sincronización bidireccional.** El Gemelo Digital debe mantener una comunicación con su contraparte real en ambos sentidos.
- **Convergencia.** Sincronización entre el gemelo y su sistema. Si por alguna razón el sistema deja de estar sincronizado, la situación debe ser detectada y corregida.
- **Verificación y validación.** La plataforma debe permitir realizar pruebas y validar los datos ingresados.
- **Soporte de tiempo real.** Las actualizaciones de los datos deben ser en tiempo real para poder afirmar que se trata de una copia virtual de un paciente real. En nuestro caso, la comunicación en tiempo real debe darse entre los *wearables* y el gemelo digital, y a su vez es requerida comunicación en *soft real-time*[1] para mostrar los datos en el panel de control del médico.
- **Interoperabilidad de plataformas.** Refiere a la extensión de un gemelo digital existente usando otros servicios que agreguen valor, tales como *Machine Learning*, visualización y simulación.
- **Abstracción de la implementación.** Permite a expertos en el dominio operar un gemelo digital sin necesitar conocimiento de la implementación técnica de la plataforma.
- **Seguridad de la información.** Previene accesos no autorizados y protege el flujo de información. Esto es de suma importancia en cualquier aplicación, pero aún más si se trata de datos de salud.
- **Modelo modificable.** Permite que si la contraparte física cambia (p. ej., la aparición de un nuevo dato que se quiera medir en los pacientes), el gemelo digital pueda cambiar y adaptarse.
- **Reusabilidad.** Es importante que la solución pueda ser agrupada en componentes y instalada en distintos centros de salud.
- **Provisionamiento.** El sistema debería poder instalarse tanto en ambientes remotos de la nube como en servidores locales.

[1] El sistema intenta cumplir con los plazos, pero no sufre fallos catastróficos si ocasionalmente no lo logra.

Dado que la mayoría de casos de uso en los que se pueden emplear Gemelos Digitales comparten características en común, crear y mantener una implementación propia puede ser considerado ineficiente.

2.3.2.2. Plataformas de Gemelos Digitales

Como mencionan Lehner y cols. (2022), una plataforma de Gemelos Digitales es una solución existente para la implementación de Gemelos Digitales que cumple con la mayoría de los requisitos básicos mencionados en la sección 2.3.2.1.

En el artículo se analizan tres plataformas distintas —Amazon, Eclipse y Microsoft DT— evaluándolas según los requisitos establecidos. En la tabla 2.1 se destacan las principales características de cada plataforma, mientras que a continuación se presenta un análisis más detallado de las diferencias entre ellas en función de los requisitos propuestos por los autores.

En base a la comparación que realizan los autores entre se establece que las tres brindan soporte básico para implementar sincronización bidireccional, pero para ponerlas en marcha se requiere una gran cantidad de programación manual.

Con respecto a la convergencia, se menciona que ninguna de las tres plataformas brinda soporte a esto.

El nivel de soporte al requerimiento de verificación y validación es distinto en cada plataforma. Amazon provee un conjunto de pruebas completo para funcionalidades específicas; Microsoft solo permite detectar automáticamente las estructuras inválidas, y Eclipse Ditto no brinda ninguna funcionalidad de verificación.

El comportamiento en tiempo real es ofrecido a medias por todas las plataformas, que ofrecen capacidades de tiempo real estricto (*hard real-time*) pero solo en dispositivos que procesan datos localmente y no en la nube. Para que el requerimiento se cumpla completamente, sería necesario que la plataforma ofreciera capacidades para medir y evaluar las restricciones de tiempo real flexible (*soft real-time*) en la nube.

Con respecto a proveer una buena abstracción, Microsoft y Eclipse proveen lenguajes para describir Gemelos Digitales y ofrecen herramientas gráficas para crearlos basadas en esos lenguajes. Amazon no ofrece esa asistencia.

La seguridad es provista por todas las plataformas ya que brindan soporte para autenticación y autorización, y la información en las comunicaciones está encriptada.

La existencia de un modelo modificable es soportada por Microsoft y Eclipse gracias a la adaptación de gemelos en tiempo de ejecución mediante interfaces. Amazon ofrece parcialmente mediante despliegues nuevos de código.

La reusabilidad se provee por las tres plataformas, mientras que el provisionamiento está brindado por Amazon y Microsoft, que dan la posibilidad de desplegar en la nube así como también herramientas para configurar integración y despliegue continuos. Eclipse no ofrece esta posibilidad.

En conclusión, la mayoría de requerimientos están cubiertos en todos los casos pero igualmente hay espacios de mejora para los servicios. Todas las plataformas estudiadas trabajan bien en cuanto interoperabilidad, seguridad y reusabilidad.

#	Plataforma	Características
1	Eclipse Ditto	▪ Ofrece APIs web para simplificar cargas de trabajo. ▪ Microservicios con el almacén de datos. ▪ Gestión de metadatos estáticos. ▪ Protocolo de comunicación basado en JavaScript Object Notation (JSON). ▪ Plataforma de código abierto.
2	Azure	▪ Modelos de dominio personalizados usando el Lenguaje de Definición de Gemelos Digitales. ▪ Representación de entornos de ejecución en vivo utilizando gráficos en tiempo real. ▪ Integración con Azure IoT Hubs y LogicApps, APIs REST para la entrada del sistema. ▪ Información procesable usando Azure Data Analytics. ▪ Alta seguridad y escalabilidad. ▪ Solución paga, aproximadamente doscientos dólares al mes.
3	AWS	▪ Conectores integrados para un acceso más rico a los datos. ▪ Gráficos avanzados de Gemelos Digitales para definir relaciones. ▪ Vista interactiva en 3D con escenas, espacio de trabajo y recursos. ▪ Complementos para Grafana para paneles de visualización personalizados. ▪ Solución paga, tiene una versión de prueba durante doce meses.

Tabla 2.1: Comparación de plataformas y sus características

2.4. Proyectos de grado antecesores

Este proyecto de grado continúa el trabajo de otros dos proyectos previos, ambos enfocados en el dominio de Dosificación de Precisión. Esta sección destaca los puntos clave de estos proyectos previos para proporcionar el contexto necesario requerido para comprender la continuación descrita en este documento.

2.4.1. Finglix: Plataforma para dosificación de precisión informada por modelos

El primero de estos, Plataforma para dosificación de precisión informada por modelos (Drocco y cols., 2021), se centra en la creación de una plataforma para el control de las dosis a administrar de ciertos medicamentos. Esta plataforma permite gestionar pacientes, realizar simulaciones y predecir dosis óptimas utilizando tanto modelos farmacocinéticos poblacionales como datos individuales de pacientes.

El proyecto integra productos de la suite Monolix de Lixoft, particularmente dos herramientas:

- **Monolix.** Es una herramienta avanzada para la estimación de parámetros en modelos no lineales de efectos mixtos. Monolix se utiliza para desarrollar modelos farmacocinéticos que estudian la variabilidad interindividual en las concentraciones plasmáticas de los fármacos.
- **Simulx.** Es una herramienta para la simulación de modelos farmacocinéticos, que permite obtener predicciones basadas en modelos y datos específicos.

El proceso que se desarrolló en el marco de esta tesis incluye la estimación de parámetros poblacionales utilizando Monolix con datos de pacientes que han recibido el fármaco previamente. Si un paciente no tiene datos previos, se utilizan parámetros poblacionales para predecir la dosificación. Además, se utiliza Simulx para realizar simulaciones donde se ingresa manualmente la información necesaria para predecir los resultados del tratamiento.

En la solución implementada por el grupo, la que llamaron Finglix, el servidor se construyó utilizando Python y Django, mientras que la web se desarrolló con React y TypeScript. La comunicación con Monolix y Simulx se gestionó mediante R utilizando el paquete LixoftConnectors.

2.4.2. Finglix 2.0: Plataforma de dosificación de precisión informada por modelos

El segundo proyecto de grado que se desarrolló en el marco de esta temática se trata de Finglix 2.0 - Plataforma de dosificación de precisión informada por modelos (Araujo, Vigna, y Betarte, 2024). Este proyecto tiene como objetivo

generalizar y adaptar Finglix para que se pueda usar con múltiples plataformas de simulación, más allá de MonolixSuite. Además, se busca aumentar la cantidad de modelos y fármacos soportados, y agregar nuevas funcionalidades que mejoren la usabilidad general de la plataforma.

Capítulo 3

Relevamiento y análisis

En este capítulo se detalla la recopilación de requisitos para la nueva plataforma. El proceso se basa en tres puntos clave. Primero, se hace un análisis del sistema previo, Finglix, que se desarrolló en en el marco de otro proyecto de grado. Para esto existe comunicación con dicho grupo de proyecto mediante las tutoras como intermediarias y luego mediante reuniones directamente con ellos. Segundo, se llevan a cabo reuniones con el equipo de la Facultad de Química, quienes brindan información sobre el uso de Finglix y proponen posibles mejoras y nuevas características. A su vez, explican sus necesidades y flujos de información que ellos consideran útiles y que serían de gran ayuda si fueran implementados. En consecuencia de esta reunión, se estudian herramientas de simulación de modelos, mencionadas por el equipo de la Facultad de Química, para considerar la posibilidad de agregarlas a la nueva plataforma.

3.1. Análisis de necesidades

En base a la recopilación de información de los distintos actores mencionados al inicio de este capítulo, se resume la información reflejándola en las necesidades de los usuarios y proveedores de la plataforma.

Finglix es la plataforma existente que está en uso por parte del Hospital de Clínicas, pero es una plataforma con cierto grado de dificultad para ser usada por usuarios médicos. En general, es usada directamente por químicos ya que se deben ingresar covariables específicas y además la interfaz no es del todo amigable para una persona que nunca la haya usado previamente. Por lo tanto, la primera necesidad es tener una plataforma más amigable para los médicos independientemente de si la había usado previamente o si posee conocimiento acerca del modelado y simulación en la dosificación de drogas. El médico debería poder ver datos del paciente, simular con ellos e ingresar información del paciente de forma sencilla y rápida.

Además, según se menciona por parte del equipo de Facultad de Química, se precisa un seguimiento del tratamiento que se le inicia a un paciente luego

de haber simulado y elegido la dosis por parte del médico. Por esto, el profesional médico debe poder simular las veces que sean necesarias, elegir la dosis e intervalo que le aplicará al paciente e iniciar el tratamiento. Luego, durante el tratamiento lo puede volver a consultar y cuando desee lo puede finalizar.

Se decide que sea una plataforma fácil de adaptar a la integración de nuevos datos (proveedores de datos, proveedores de información, modelos) por lo que el médico debe poder elegir con qué proveedor de simulación simular dentro de los existentes. A su vez, un administrador debe poder configurar nuevos datos a integrar como modelos o proveedores desde un panel de administrador. Como base, se deben integran Finglix y Mrgsolve, pero luego facilitar la integración si se requiere algún otro.

Otra necesidad planteada por el equipo de Facultad de Química es la de poder tener los datos del paciente actualizados en todo momento, para poder ejecutar las simulaciones de los modelos con los datos más recientes posibles y así obtener una simulación más personalizada y actualizada. Para esto, se deben vincular los datos ingresados manualmente de cada paciente con las covariables de cada simulación automáticamente y que el médico las pueda editar si lo desea. También, al querer llevar un diagnóstico en tiempo real del paciente, se requiere la vinculación de la plataforma con dispositivos que midan datos del paciente y en tiempo real los envíen a la aplicación para que el médico los pueda ver y utilizar en las simulaciones.

Al ser una plataforma médica que maneja datos sensibles, se requiere que tenga seguridad en el intercambio de información y en la autenticación de usuarios. Por lo tanto debe tener un proceso de *login* para médicos y pacientes seguro, como p. ej. mediante el usuario Gub.uy (AGESIC, 2022).

Finalmente, para maximizar las ventajas de aplicar el concepto de Gemelos Digitales en la medicina, se establece que la plataforma debe ser extensible a otros dominios además del de dosificación, como p. ej., nutrición. Además, se requiere la capacidad de integrar múltiples fuentes de datos para garantizar que la información disponible sobre los pacientes sea lo más completa posible, con el fin de tener un mayor grado de personalización en los tratamientos. También se debe contemplar la posibilidad de integrar otros proveedores de modelos de simulación para que la plataforma se adapte las necesidades de distintos centros de salud.

3.2. Requerimientos

En esta sección se listarán los requerimientos de la plataforma a crear. Para la implementación de la obtención de información del paciente en tiempo real, se decide hacer una aplicación móvil que pueda obtener toda la información de salud del paciente de su celular y enviarla directamente a la base de datos de la plataforma para que el médico la pueda visualizar y consultar.

3.2.1. Requerimientos Funcionales

Se tienen requerimientos funcionales según el rol del usuario que desea interactuar con la plataforma. El rol de administrador y el rol de médico deben poseer las siguientes funcionalidades.

- **Administrador**:
  1. Integrar un nuevo proveedor de datos si corresponde.
  2. Integrar un nuevo proveedor de simulación si corresponde.
  3. Integrar un nuevo modelo de simulación para un proveedor dado.
  4. Crear usuarios de tipo Médico en el sistema.
  5. Editar usuarios de tipo Médico en el sistema.
  6. Borrar usuarios de tipo Médico del sistema.
  7. Crear usuarios de tipo Paciente en el sistema.
  8. Editar usuarios de tipo Paciente en el sistema.
  9. Borrar usuarios de tipo Paciente del sistema.
- **Médico (requisitos generales de manejo de datos)**:
  1. Ingresar al sistema autenticándose mediante la plataforma Gub.uy. El médico deberá acceder al sistema mediante credenciales que confirmen su identidad.
  2. Cargar datos y observaciones del paciente de forma manual.
  3. Asociar teléfono móvil a un paciente para recabar datos en tiempo real.
  4. Visualizar historial de datos del paciente incluyendo datos obtenidos de proveedores como los utilizados para las simulaciones realizadas para el paciente.
  5. Visualizar datos en tiempo real del paciente.
  6. Elegir qué datos de *wearables* observar del paciente dentro de los soportados por la plataforma.
  7. Cerrar sesión.
- **Médico (requisitos específicos de solución)**:
  1. Acceder al dominio de dosificación. Tener acceso a un sector de la plataforma dedicado especialmente a la dosificación de precisión.
  2. Realizar simulaciones con Finglix como proveedor de las mismas utilizando sus modelos.
  3. Realizar simulaciones con Mrgsolve como proveedor de las mismas utilizando sus modelos.

4. Poder simular la concentración de las drogas en función del tiempo para un paciente dado. Estas drogas serán las pertenecientes a los proveedores de simulación cargados en la plataforma.
5. Utilizar datos del paciente, sea cual sea su fuente, para utilizarlos como entrada al simular con las drogas.
6. Visualizar los resultados de cada simulación creada (Datos de entrada así como datos obtenidos de la gráfica resultante).
7. Realizar múltiples simulaciones por droga y vincularlas a un mismo tratamiento. Poder visualizar todas las simulaciones por tratamiento al mismo tiempo en la misma gráfica para comparar y tomar decisiones.
8. Iniciar tratamiento seleccionando una simulación realizada como la que se le aplicará al paciente.
9. Visualizar tratamientos, sus simulaciones asociadas, sus fechas de inicio y fin (si está activo o finalizado) y la simulación elegida si es que está activo.
10. Visualizar los tratamientos activos y poder finalizarlos desde la pantalla principal.
11. Visualizar tratamientos clasificados según la droga utilizada para visualizar la evolución de las dosis utilizadas para el paciente.
12. Finalizar tratamiento.

- **Paciente**:

1. Ingresar a la aplicación mediante Gub.uy o *login* seguro para verificar la identidad del paciente.
2. Seleccionar los datos de salud a los que la aplicación tiene permiso de acceso.
3. Transmitir información de salud en tiempo real, incluso cuando la aplicación se encuentra cerrada o en *background*.
4. Visualización de los datos que se están transmitiendo al proveedor de salud en tiempo real.

3.2.2. Requerimientos No Funcionales

1. Utilizar *software* de código abierto.
2. Utilizar una plataforma de Gemelos Digitales.
3. Desarrollar una plataforma amigable y fácil de usar tanto para usuarios médicos como para usuarios pacientes.
4. Implementar una base de datos escalable para guardar grandes volúmenes de datos provenientes de *wearables*.

5. Proporcionar soporte para tiempo real.

6. Utilizar mecanismos de seguridad en la plataforma, en particular, con comunicaciones cifradas (TLS, etc.).

7. Desarrollar una plataforma genérica y configurable para poderla extender a futuro.

3.2.3. Aplicación a la solución

En la figura 3.1 se ilustran los requerimientos funcionales de forma gráfica para una mejor comprensión de los mismos. La ilustración explica los flujos listados anteriormente en esta sección y la interacción entre los agentes de la plataforma.

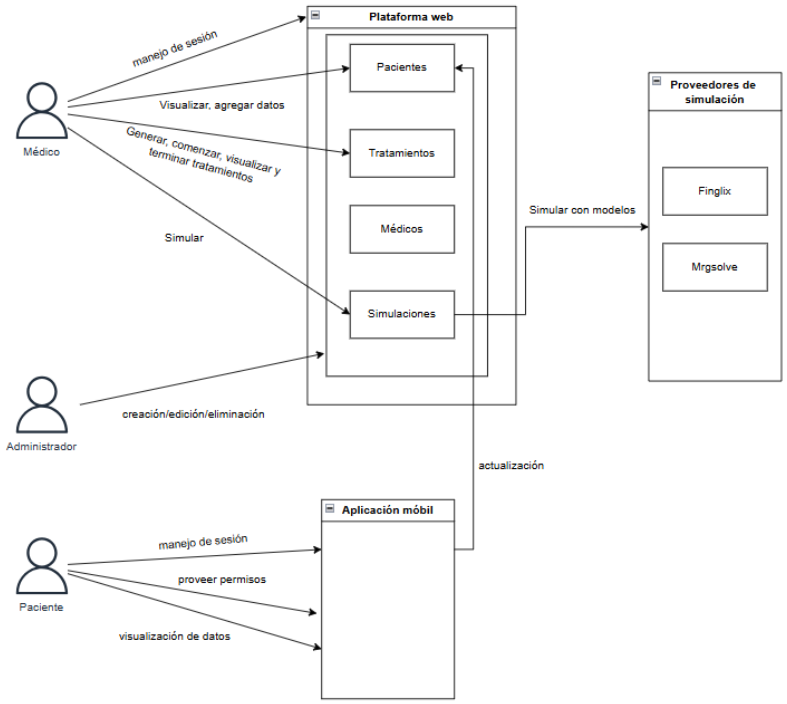


Figura 3.1: Requerimientos funcionales

3.3. Plataformas similares

En esta sección se presentan otras plataformas que cumplen parcialmente con los requisitos planteados en este proyecto. Este análisis se realiza para evaluar las características y limitaciones de cada plataforma en relación con los objetivos planteados, con el fin de identificar posibles soluciones o enfoques que puedan ser adaptados o mejorados para cumplir con las necesidades del proyecto.

3.3.1. ELEM Biotech

Esta plataforma aplica Gemelos Digitales al área médica de Cardiología. Usa información específica del paciente para crear un modelo del corazón y así poder ayudar en la creación de dispositivos médicos. Esta *start-up* ofrece a las compañías la habilidad de experimentar con drogas y dispositivos en modelos simulados de corazones humanos, con el objetivo de ayudar a los cirujanos a planear las cirugías para pacientes con latidos irregulares o con fibrilación articular. Hacer predicciones específicas del paciente ayuda a saber las consecuencias a largo plazo y a decidir qué válvula cardíaca usar y, p. ej., dónde insertarla durante un trasplante de la misma (ELEM Biotech SL, 2015).

3.3.2. GSK Drug Company

Según el artículo de Kollewe (2021), expertos en inteligencia artificial trabajan en crear una réplica digital de tumores de pacientes para ayudar al tratamiento de cáncer. Utilizan imágenes e información genética y molecular como también las células cancerígenas en tres dimensiones del paciente. Aplicando aprendizaje automático a esta información, los científicos pueden predecir cómo va a responder un paciente a cierta droga, combinación de drogas o a cierta dosis. Esto no se puede hacer repetidas veces con el paciente real porque cada vez que se prueba con una droga se estaría haciendo un ensayo clínico distinto.

Esta investigación incluye a diez expertos en la creación de la inteligencia artificial de la droga y diez oncólogos especialistas del King´s College de Londres. Trabajan en la razón de por qué solo un quinto de los pacientes responden de buena forma a tratamientos inmuno-oncológicos. Primero quieren desarrollarlo para cancéres más específicos y después poder ampliarlo a un marco en el que otras personas puedan contribuir.

Un profesor del King´s (Kollewe, 2021) argumenta que en general la mitad de pacientes que tienen cáncer avanzado pero tratable, volvieron uno o dos años después cuando se les descubrió que se esparció a otras partes del cuerpo. Este avance tecnológico podría reducir el número de los pacientes que se vean afectados por esto.

3.3.3. Exact Cure

Exact cure es una compañía situada en Francia que, mediante inteligencia artificial y una base de datos de pacientes, hace simulaciones de efectos y dosis

de drogas para pacientes. Se enfoca en mejorar los resultados de los tratamientos mediante un procedimiento personalizado. Cuanta más información personal se le brinde, tanto la básica (edad, sexo, peso, altura) como la específica (análisis de sangre, perfil genético, función renal, función del hígado), mejor será la predicción.

Se puede monitorear a familiares y compartir el perfil de usuario con otros. Está ideada tanto para pacientes como para médicos, laboratorios y proveedores de seguros de salud. En la página se tiene como ejemplo una simulación en la que ingresando datos se obtiene un gráfico de la ventana terapéutica óptima del paracetamol (Exactcure, 2020).

3.3.4. Conclusiones del análisis de plataformas similares

Si bien todas estas plataformas presentan avances innovadores que aplican el concepto de Gemelos Digitales al área de la salud, se concluye que ninguna cumple de forma completa con todos los requerimientos descritos en secciones anteriores. Si bien algunas permiten realizar simulaciones en base a datos recolectados mediante el uso de un gemelo digital, como Exact Cure, ninguna brinda una interfaz con los datos del paciente en tiempo real para el médico, o la posibilidad de gestionar tratamientos. Este proyecto toma diversas ideas basadas en este análisis, tales como el ingreso de datos manuales que brinda Exact Cure o el concepto de una figura humana virtual introducido por ELEM Biotech.

3.4. Definición del alcance

Dado que se trata de un proyecto de grado, se decide acotar el alcance y mantener únicamente las funcionalidades que se consideraron cruciales para una solución integral, con el fin de cubrir la necesidad de la mejor manera posible. Es de esta forma que el compromiso final comprende todos los requerimientos funcionales, excepto:

- **Integrar un *plugin* con otro proveedor de datos si corresponde.** Se consideraron los beneficios de integrar la historia clínica, pero se priorizaron otros requisitos.
- **Autenticación mediante la plataforma Gub.uy.** El análisis realizado concluye que esta integración sería costosa y, por el momento, puede ser sustituida por métodos de *login* tradicionales.

Capítulo 4

Diseño de la solución

En base al análisis realizado en el capítulo previo, en este capítulo se presenta la propuesta de solución planteada para el proyecto. En particular, se provee una descripción general de la solución y se detallan los componentes que la conforman.

4.1. Descripción de la solución: MedFing

La plataforma propuesta, MedFing, busca resolver la problemática planteada proporcionando una interfaz para que los profesionales de la salud puedan monitorear en tiempo real los datos de salud de sus pacientes, integrando información de dispositivos móviles y otros sistemas de salud. Además, permite realizar simulaciones personalizadas para evaluar la evolución de tratamientos y ajustar las dosis de medicamentos de manera precisa, optimizando así los resultados clínicos y mejorando la calidad de atención.

En la figura 4.1 se muestra el diagrama de clases de la solución planteada. La entidad **Médico** modela al profesional de salud que, mediante un usuario, puede acceder a las funcionalidades de la plataforma. Uno o varios médicos supervisan a un grupo de entidades del tipo **Paciente**. Los modelos están representados por el tipo **Modelo**. Los **Proveedores de simulación** usan estos modelos para relacionarse con las entidades que representan a los pacientes y el fruto de esta relación es un objeto de tipo **Simulación**. Todos los proveedores de simulación pertenecen a un **Dominio**: dosificación es uno de los dominios posibles, pero la solución se estructura de forma que se pueda extender a otros dominios en el futuro, tales como nutrición.

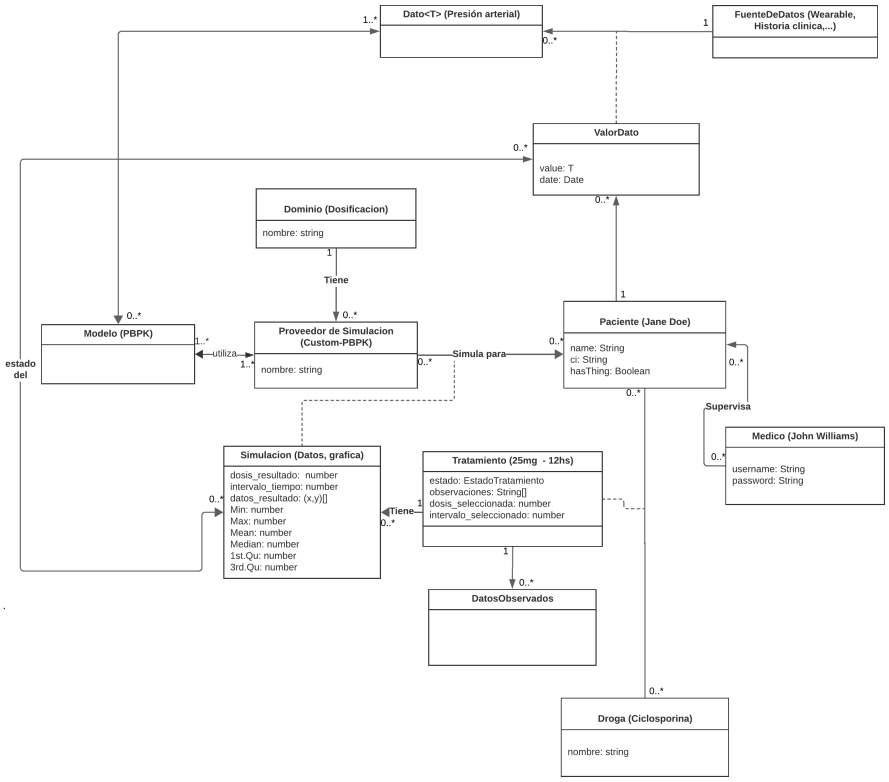


Figura 4.1: Diagrama de clases

A su vez, existe la entidad **Droga** para representar el fármaco. Esta se relaciona con los pacientes y genera un **Tratamiento**. Se elige el nombre tratamiento para el objeto que agrupa varias simulaciones con el fin de que el médico las compare y seleccione una dosis final que se le va a administrar al paciente. Los tratamientos tienen estado: pueden encontrarse en modo borrador (sin dosis final seleccionada), activos o finalizados.

Finalmente, los datos de salud del paciente se representan mediante las entidades **Fuente de datos** y **Dato**. La combinación de dos instancias de estas clases genera un objeto que llamamos **ValorDato**. Mientras **Dato** representa el tipo del dato (p. ej., presión arterial), **ValorDato** guarda el valor de la medida en una fecha. La fuente de datos representa la procedencia de ese dato. Un paciente puede tener muchos objetos **ValorDato** asociados.

4.2. Arquitectura de la solución

En la figura 4.2 se presenta un diagrama de alto nivel de la arquitectura de la solución propuesta. Este sistema cuenta con:

- **Una aplicación frontend** en la que los médicos pueden monitorear los datos de salud de los pacientes en tiempo real, así como también agregar datos, simular dosis de distintos medicamentos, crear y gestionar tratamientos y documentar observaciones.
- **Un panel de administración** que permite crear, visualizar, modificar y borrar todas las entidades soportadas por la plataforma.
- **Una aplicación móvil** que se encarga de recolectar, manejar y compartir los datos de salud de los pacientes con su proveedor de salud. Es un intermediario entre la conexión con dispositivos *wearables* y el servidor.
- **Un servidor web** con el que se comunican los tres componentes de frontend mediante una API REST y que se encarga de gestionar la lógica de negocio, manejar las solicitudes de los usuarios, y garantizar la correcta integración de los datos provenientes de diferentes fuentes. En particular, tiene la tarea de integrar distintos proveedores de simulación, tales como Finglix, y normalizar la información de manera que todas las simulaciones sigan el mismo formato sin importar el proveedor que las generó. El servidor también se encarga de coordinar la seguridad y autenticación de los usuarios.
- **Una base de datos** encargada de persistir toda la información relevante para la plataforma, como los datos de los pacientes, médicos, simulaciones, tratamientos, etcétera.
- **Un proveedor de infraestructura de Gemelos Digitales** que soporta la creación de entidades para el uso como Gemelos Digitales, comunicación bidireccional en tiempo real, protocolos de mensajería específicos de IoT, encripción de la información transmitida, entre otras cosas.
- **Un módulo proveedor de simulación** que expanda la cantidad de modelos disponibles y que permita la integración con nuevos modelos en el futuro, ilustrando la escalabilidad del producto y abriendo puertas para otras opciones de simulación que mejor se ajusten a los beneficios de los Gemelos Digitales.

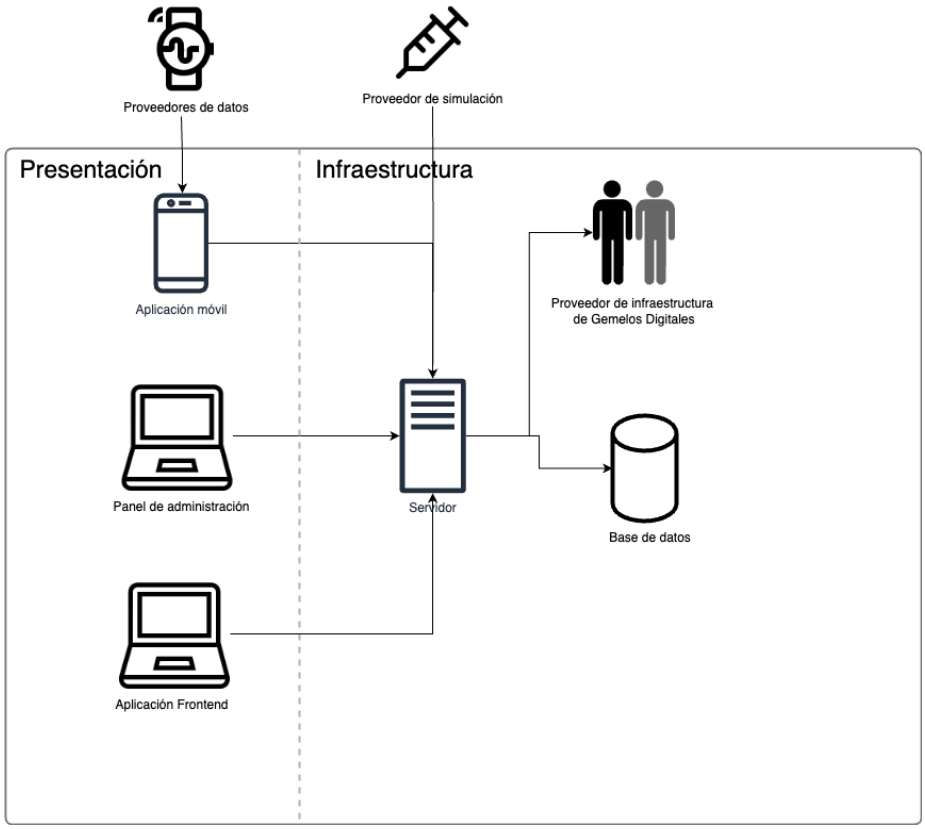


Figura 4.2: Arquitectura del sistema

4.3. Componentes de la solución

En la figura 4.3 se visualiza el diagrama de componentes para el sistema propuesto. Este tipo de diagramas consiste en una vista estática de los componentes y sus relaciones. Cada nodo es un componente que representa un conjunto de clases con una interfaz bien definida. Cada arista es una relación que representa la conexión «usa los servicios de».

El desarrollo basado en componentes (CBD) y el desarrollo orientado a objetos son desarrollos que se complementan, aunque generalmente la tecnología preferida para crear componentes es la basada en objetos. Los diagramas basados en componentes, como explica Ambler (2004), permiten modelar componentes de *software* y sus interfaces, lo que facilita la organización del esfuerzo de desarrollo al crear el *software*.

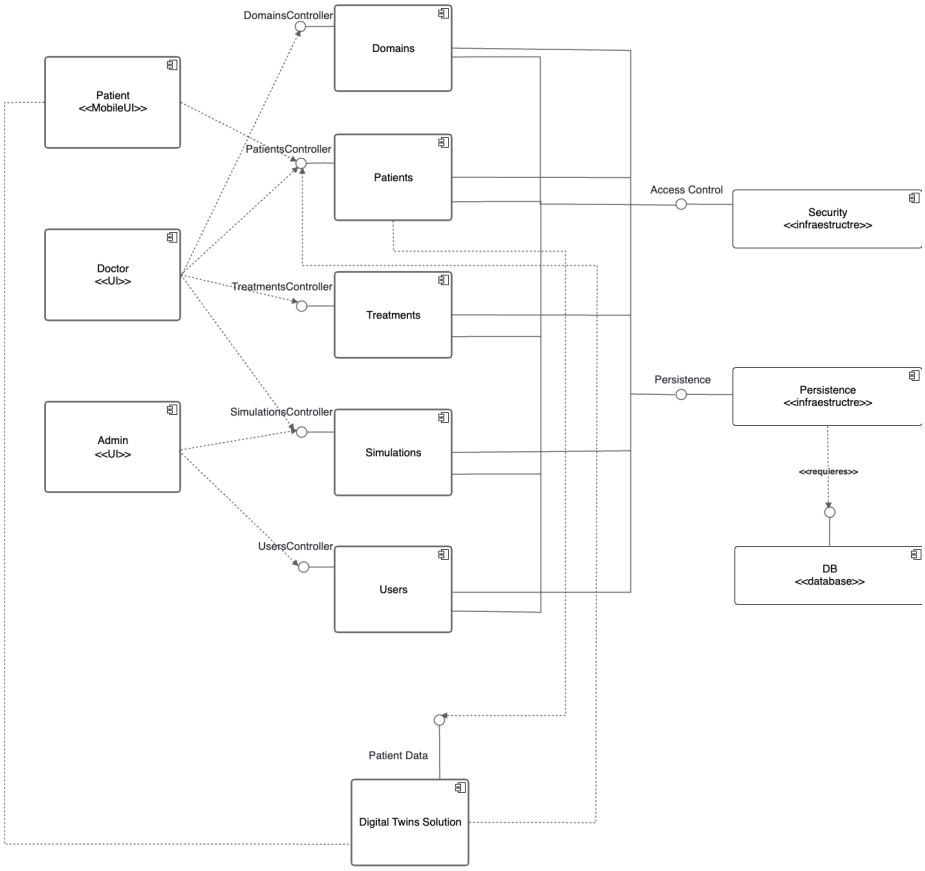


Figura 4.3: Diagrama de componentes

Los controladores gestionan la lógica relacionada con cada entidad (dominios, pacientes, tratamientos, simulaciones). Se encargan de procesar las solicitudes que llegan desde las interfaces de usuario y de brindar una capa de acceso a datos.

Se representa la integración con esta solución externa de Gemelos Digitales como otro componente, que se comunica con la aplicación móvil para recabar datos de salud de los pacientes.

En la figura 4.4 se ilustran con mayor detalle las funciones diseñadas para cada controlador, con sus correspondientes parámetros.

User Controller maneja la gestión de usuarios, incluyendo la eliminación, obtención, actualización y listado de usuarios con parámetros de paginación opcionales. Por otra parte, el **Domain Controller** está enfocado en obtener los dominios y proveedores de simulación para un dominio específico. El componente **Simulation Controller** es responsable de crear y obtener simulaciones,

User Controller

```
deleteUserById(id: number)
getUserById(id: number)
updateUser(id: number, name: string, email: string, role: number)
getUsersList(offset?: number, limit?: number)
```

Domain Controller

```
getDomains()
getSimulationProviders(domainId)
```

Simulation Controller

```
addSimulation(simulationData: SimulationData)
getSimulation(simulationId: string)
```

Patient Controller

```
getPatientByCi(ci: string, authToken: string)
returnPatientByCi(doctorId: string, ci: string)
filterToLatestValues(basicInformation: any[])
updatePatient(id: string, doctorId: string, name: string, email: string, role: string)
addManualValueToPatient(patientCi: string, typeId: number, name: string, value: any, unitOfMeasurement: string,
authToken: string)
getPatientsList(authToken: string, filter?: string, offset?: number, limit?: number)
getValuesFromThing(ci: string)
generateOtp(ci: string)
linkMobileDevice(ci: string, otp: string)
updatePatientValidator(body: { name: string; ci: string })
```

Treatment Controller

```
endTreatment(treatmentId: string)
startDraftTreatment(treatmentId: string, simulationId: string)
addSimulationToTreatment(
  modelId: string,
  treatmentId: string,
  ci: string,
  simulationData: SimulationData,
)
getTreatments(ci: string, onlyActive?: boolean)
getTreatmentsList(ci: string)
getActiveTreatmentsList(ci: string)
addTreatment(
  ci: string,
  status: string,
 treatmentData: TreatmentData,
)
```

Figura 4.4: Especificación de controladores

mientras que **Patient Controller** tiene una funcionalidad más amplia, como la gestión de pacientes a través de su cédula de identidad, brindar datos de pacientes y actualizarlos, y la vinculación con la aplicación móvil. También incluye otras funcionalidades, como la posibilidad de agregar datos manuales a un paciente. Finalmente, **Treatment Controller** gestiona los tratamientos de los pacientes, permitiendo iniciar, terminar y agregar simulaciones a los tratamientos. También ofrece métodos para obtener listas de tratamientos, ya sean todos o solo los activos, y para agregar nuevos tratamientos.

4.4. Detalles de Diseño

En esta sección se brindan detalles de diseño en cuanto a autenticación, diseño del backend y despliegue.

4.4.1. Autenticación

En esta sección se detallan las distintas formas de autenticación utilizadas en el proyecto.

4.4.1.1. JWT (Json Web Tokens)

Como se explica en su documentación oficial (JSON Web Token, s.f.), JWT (JSON Web Token) es un estándar abierto (RFC 7519) que define una forma compacta y autónoma de transmitir información de forma segura entre partes como un objeto JSON. Un JWT se compone de tres partes: un encabezado (Header), la carga útil (Payload) y la firma (Signature). El encabezado contiene el algoritmo de la firma y el tipo del token (JWT en este caso). En la carga útil se encuentran varios datos y entre ellos el emisor, el tiempo de expiración y el asunto. Por último, la firma se crea tomando el *header* y *payload* codificados, usando el algoritmo especificado y firmando con una clave secreta.

Dentro de las ventajas del uso de JWT se tiene que es autónomo, ya que contiene toda la información necesaria. Además, es lo suficientemente compacto como para mandar en un *header* de autenticación en una solicitud HTTPS, y al ser firmado digitalmente, es seguro. Es un método muy usado, por lo que se resuelve usarlo para la autenticación entre el *frontend* y *backend*.

En el caso de este proyecto, se utiliza un *token* para autenticar al usuario que ingresa a la plataforma y se guarda solamente el identificador del usuario y su correo electrónico en el cuerpo del mensaje. Se utiliza una clave secreta encriptada y guardada de forma segura. Luego, en cada consulta desde el frontend se manda este *token* como encabezado de autenticación para corroborar que la consulta proviene del usuario que dice ser y de una sesión válida. Para esto, al autenticarse un usuario, se guarda el *token* del mismo y así se podrá mandar en las siguientes solicitudes HTTPS.

Además, el *token* de *login* se guarda en el navegador de forma que, cuando el usuario entra a la aplicación, se hace un *login* automático (si el *token* no ha

vencido) o se genera un *token* nuevo (si ha vencido el previo).

Siguiendo las las buenas prácticas del JWT, se utiliza HTTPS siempre, se establece un tiempo de expiración adecuado, no se envía información sensible en el *payload*, se almacena de forma segura y se usan los *refresh tokens* para la renovación y autenticación automática.

4.4.1.2. Autenticación con OTPs

Para que la aplicación pueda ser usada por pacientes reales, es necesario que exista un proceso de verificación de la identidad de los usuarios móviles. Se evalúan distintas opciones, pero la decisión final consiste en la implementación de un sistema de validación de identidad mediante un código OTP de cuatro dígitos que el médico pueda proporcionar al paciente durante la consulta, y de esta manera vincular la aplicación móvil con el usuario creado en el sistema. El flujo implementado se puede ver en el diagrama de secuencia representado en la figura 4.5.

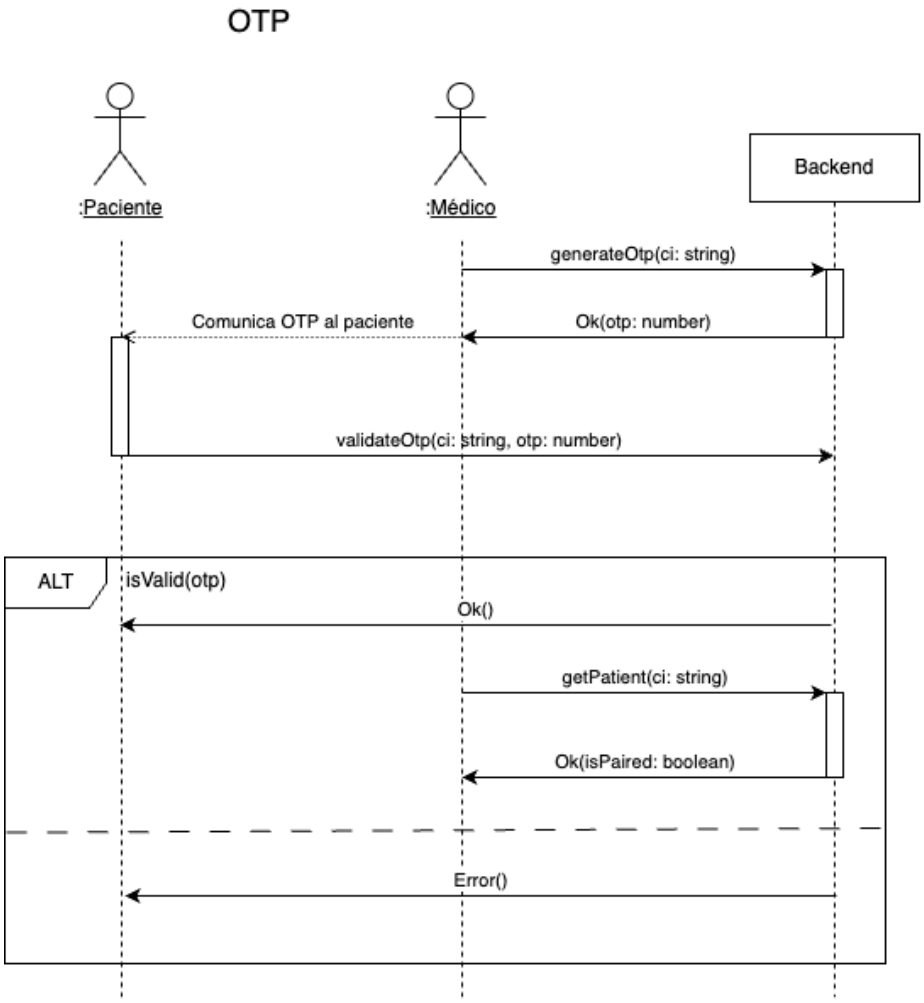


Figura 4.5: Diagrama de secuencia para verificación mediante OTP

1. **Generación del OTP**: Cuando el paciente se encuentra en consulta, el médico solicita el emparejamiento desde la aplicación web y el sistema genera un código OTP de cuatro dígitos, único para ese paciente y esa sesión. Este código puede ser visualizado por el médico.

2. **Entrega del OTP al paciente e ingreso en la aplicación**: El médico proporciona este código OTP al paciente durante la consulta. Este código, en conjunto con la cédula de identidad, sirve como medio de emparejamiento entre el perfil del paciente en el sistema y la aplicación móvil que el paciente tiene instalada en su dispositivo.

3. **Validación del OTP**: Una vez ingresado, la aplicación móvil envía el OTP al sistema para su validación. El sistema compara el código recibido con el generado anteriormente. Si coinciden, el proceso de verificación es exitoso y se le informa al usuario médico y al usuario paciente.

4.4.2. Diseño del backend

En esta sección se detallarán los conceptos y convenciones utilizadas al diseñar la arquitectura del backend y se explicarán las decisiones tomadas.

4.4.2.1. RESTful API

REST (Representational State Transfer), como se explica en el artículo (Amazon, 2023), define un conjunto de estándares para servicios web. Una API es una interfaz que programas de *software* utilizan para comunicarse entre ellos. Por lo que una API RESTful es una API que sigue los estilos y estándares de una arquitectura REST. Los sistemas REST son escalables, sin estado, almacenables en caché y que tienen una interfaz uniforme.

4.4.2.2. CRUD API

Al crear una API, se quiere que el modelo provea cuatro funcionalidades básicas. Debe ser capaz de crear, leer, modificar y eliminar recursos. Este conjunto de operaciones esenciales es llamado CRUD. En el artículo (Amazon, 2023) se menciona que en general, las RESTful APIs utilizan solicitudes HTTP. En una API REST, cuatro de los métodos HTTP más comunes son: GET, POST, PUT y DELETE, que son métodos mediante los cuales un desarrollador puede crear un sistema CRUD.

Se usa el HTTP POST para crear un recurso en el ambiente. Asimismo, se usa el HTTP GET para leer un recurso, sacando información sin alterarla. Si se busca modificar, se usa el HTTP PUT para actualizar un recurso. Finalmente, se utiliza el HTTP DELETE para eliminar un recurso del sistema.

4.4.2.3. Convenciones RESTful para rutas

Las rutas en la API se pueden definir utilizando las convenciones RESTful, que dictan la estructura de las URL y el uso de los métodos HTTP para que la API sea más intuitiva y fácil de usar. Algunos ejemplos de estas convenciones son:

- GET `/treatments`: Para obtener una lista de tratamientos.
- POST `/treatment`: Para crear un nuevo tratamiento.
- PUT `/treatment-:treatmentId/simulation/:simulationId/start`: Para comenzar un tratamiento con una simulación específica.

Esta organización permite que las rutas sean predecibles y consistentes.

4.4.2.4. División en Rutas por Entidad y Componente

Para modularizar la API se divide en diferentes partes, cada una encargada de una entidad específica del sistema (p. ej., Pacientes, Tratamientos, etc.). Cada entidad tiene su propio conjunto de rutas, es decir, los *endpoints* que se usan para interactuar con los recursos de la base de datos. Ejemplo: GET `/tratamientos`, POST `/tratamiento`. A su vez, cada entidad tiene su propio controlador, que son los encargados de manejar la lógica de la aplicación, como procesar las solicitudes HTTP, validar los datos o manejar errores. Por último, cada entidad tiene su capa de acceso a datos, que se encarga de realizar las operaciones de acceso a la base de datos (las consultas SQL, en este caso). De esta manera, los controladores no tienen que lidiar directamente con la base de datos.

Este diseño sigue el principio de separación de intereses, donde cada capa de la aplicación tiene una responsabilidad específica, lo que facilita su comprensión y modificación sin afectar otras partes del código.

4.4.2.5. Diseño elegido

Se decide crear una CRUD RESTful API permitiendo las operaciones CRUD (GET, POST, PUT, DELETE). La API seguirá una estructura modular, basada en la separación de responsabilidades, que es fundamental para crear código escalable, mantenible y verificable. Para esto, se seguirán las convenciones de una API RESTful y la división de rutas por entidad y componente.

4.4.3. Despliegue

Según Ambler (2004), los diagramas de despliegue de UML muestran una vista estática de la configuración de los nodos de procesamiento y los componentes que se ejecutan en esos nodos. Es decir, muestran el hardware para el sistema, el software instalado en ese hardware y el middleware utilizado para conectar las máquinas entre sí. Se crea un diagrama de despliegue para aplicaciones que se despliegan en varias máquinas. También se pueden crear diagramas de despliegue para explorar la arquitectura de sistemas integrados, mostrando cómo funcionan juntos los componentes de hardware y software.

En la figura 4.6 se ilustra un diagrama de despliegue del sistema diseñado en este documento.

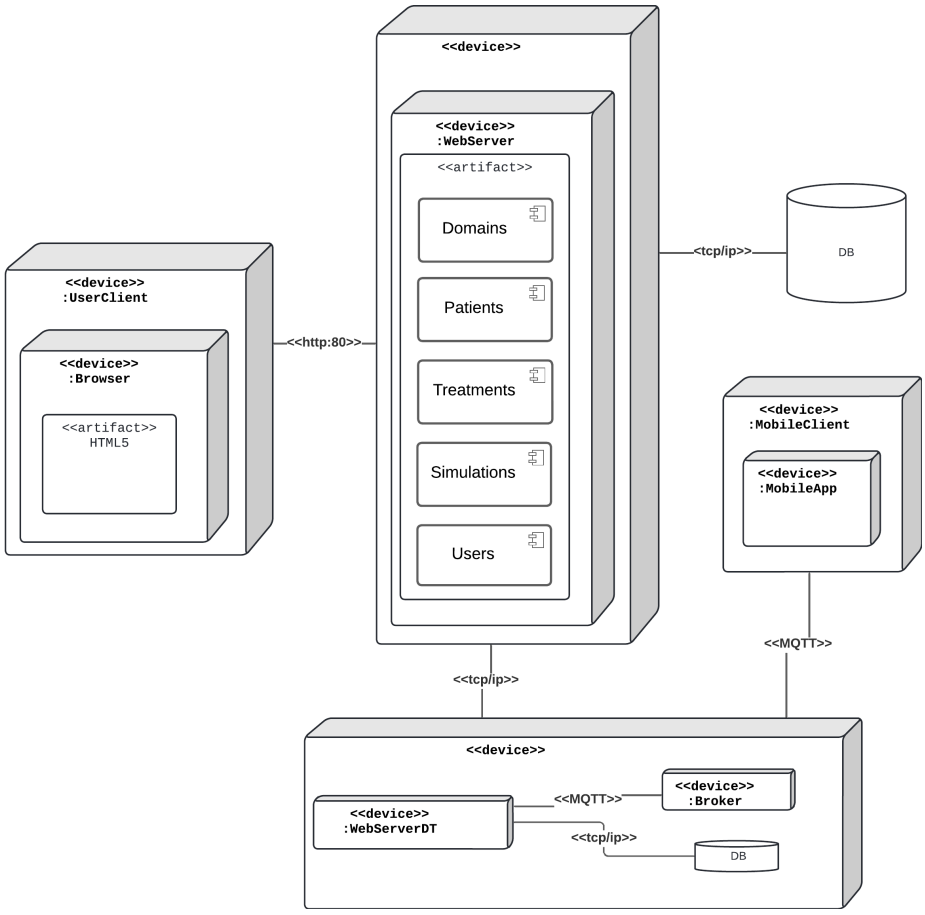


Figura 4.6: Diagrama de despliegue

4.5. Principales interacciones

En esta sección se describen las principales interacciones de los usuarios con la plataforma. Un flujo importante es la interacción del paciente y doctor con el proveedor de gemelos digitales. En la figura 4.7 se puede visualizar un diagrama de secuencia que ilustra esta interacción.

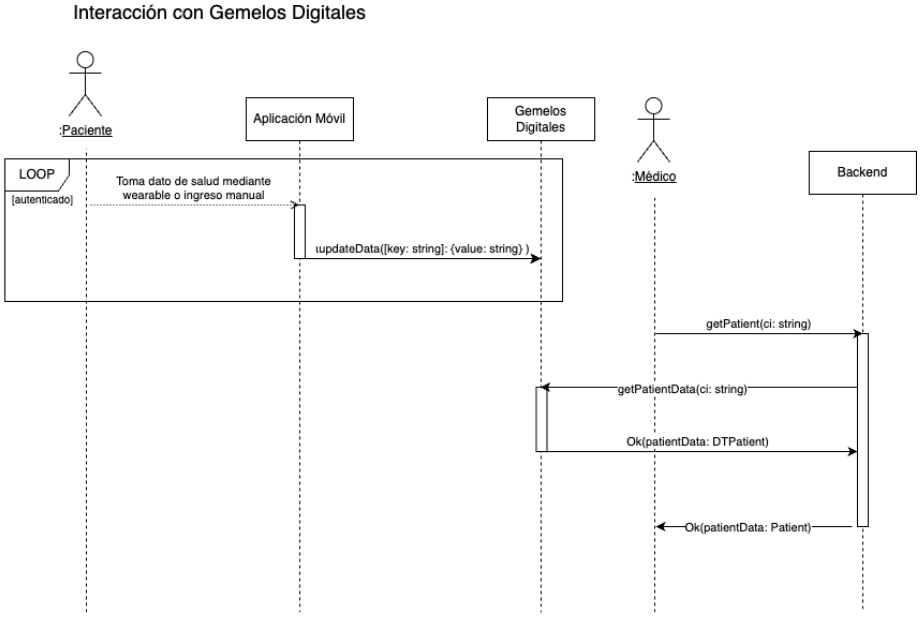


Figura 4.7: Diagrama de secuencia de la interacción con el proveedor de Gemelos Digitales

1. **Paciente registra datos en la aplicación móvil:**
   - El paciente, estando autenticado, ingresa datos de salud manualmente o mediante un dispositivo *wearable*. El *loop* alrededor de la interacción del paciente indica que este flujo (toma de datos) puede repetirse mientras el paciente esté autenticado.
   - La aplicación móvil realiza una solicitud `updateData({key: string}: {value: string})` al sistema de Gemelos Digitales para registrar estos datos.
2. **Doctor solicita datos del paciente:**
   - El doctor inicia la interacción solicitando información del paciente al *backend*.
3. **Backend consulta a proveedor de Gemelos Digitales:**
   - Para responder al doctor, el *backend* envía una solicitud `getPatientData(ci: string)` al proveedor de Gemelos Digitales, pasando el número de cédula del paciente.
4. **Backend devuelve los datos del paciente:**

- Luego de obtener los datos del proveedor de Gemelos Digitales, el *backend* responde con la información del paciente al doctor.

Otra interacción importante se trata del flujo de simular una dosis. En la figura 4.8 se puede visualizar el diagrama de secuencia para este flujo.

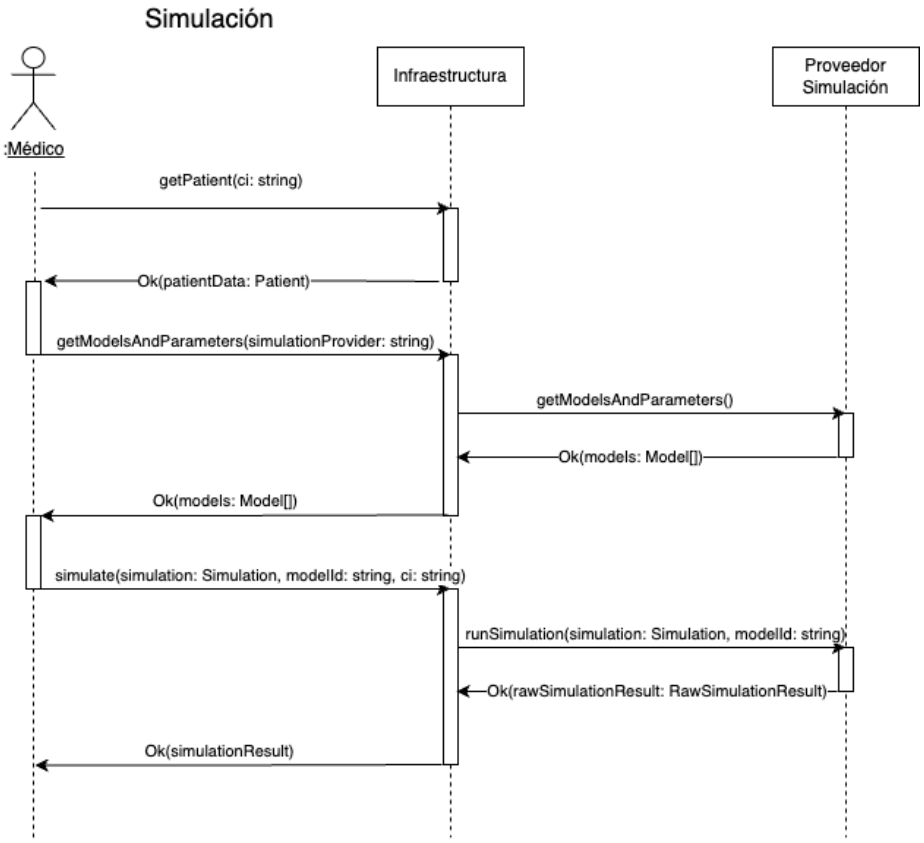


Figura 4.8: Diagrama de secuencia de la interacción con el proveedor de Gemelos Digitales

1. **Obtener datos del paciente:**
   - El doctor solicita los datos del paciente enviando una solicitud `getPatient(ci: string)` a la **Infraestructura**.
   - La **Infraestructura** responde con `Ok(patientData: Patient)`, proporcionando la información del paciente.
2. **Obtener modelos y parámetros:**

- El doctor solicita a la **Infraestructura** los modelos disponibles para el proveedor de simulación deseado mediante la función `getModelsAndParameters`.
- La **Infraestructura** envía la solicitud `getModelsAndParameters` al **Proveedor de Simulación**.
- El **Proveedor de Simulación** responde con una lista de modelos `Ok(models: Model[])` que luego la **Infraestructura** reenvía al doctor.

3. **Ejecutar simulación:**
   - El doctor solicita ejecutar una simulación enviando `simulate(simulation: Simulation, modelId: string, ci: string)` a la **Infraestructura**.
   - La **Infraestructura** reenvía la solicitud `runSimulation(simulation: Simulation, modelId: string)` al **Proveedor de Simulación**.
   - El **Proveedor de Simulación** ejecuta la simulación y responde con `Ok(rawSimulationResult: RawSimulationResult)`.
   - La **Infraestructura** procesa el resultado y envía `Ok(simulationResult)` al doctor.

Capítulo 5

Implementación

En este capítulo se describen las decisiones de implementación tomadas en cada parte que compone la solución global y su respectiva justificación.

La solución consiste en una aplicación web (*frontend*) construida con ReactJS, conectada mediante una API REST con su respectivo servidor (*backend*) creado en Node.js y con una base de datos PostgreSQL.

A su vez, la solución incluye una aplicación móvil creada en React Native y conectada con Health Connect para la interacción con los datos en tiempo real del paciente. Esta aplicación se conecta mediante una conexión de protocolo MQTT al *framework* Eclipse Ditto, donde se almacenan los datos del gemelo digital del paciente. También se conecta al servidor, con el fin de realizar la validación de identidad de cada paciente.

La aplicación web y el servidor se comunican con Eclipse Ditto, tanto para extraer datos como para crear y conectar cada paciente. Finalmente, se crea un panel de administrador usando una integración de Node.js y AdminJS. La figura 5.1 muestra un diagrama similar al de arquitectura pero con los componentes relacionados a las tecnologías con las que son implementados en el proyecto.

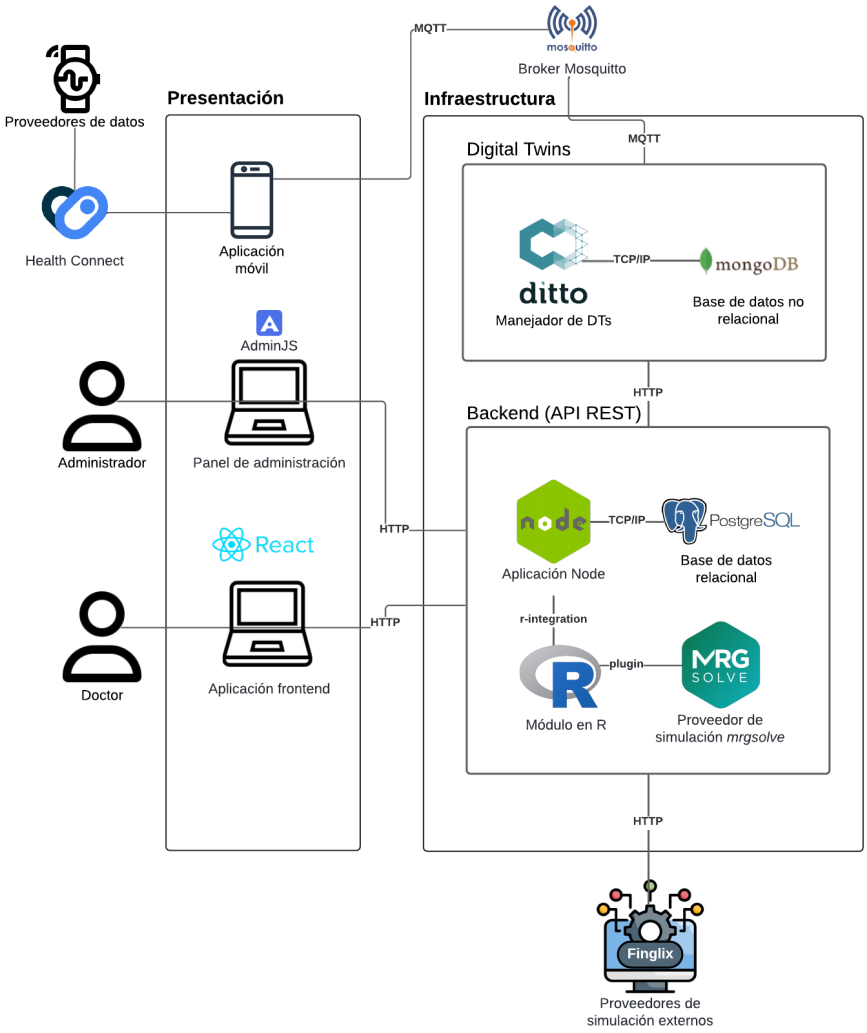


Figura 5.1: Diagrama de implementación

5.1. Infraestructura

En esta sección se detallan las decisiones tomadas con respecto a la implementación de las distintos componentes que forman parte de la capa de infraestructura de la plataforma.

5.1.1. Backend

En esta sección se establecen las decisiones de implementación tomadas a la hora de desarrollar el *backend*.

5.1.1.1. Conceptos y stack tecnológico

A continuación se detallan conceptos y definiciones clave para las decisiones de diseño acerca del *backend*. Se buscan distintas fuentes de buenas prácticas para la arquitectura del mismo y se elige el artículo (CRUD REST API with Node.js, Express, and PostgreSQL, 2024) debido a su claridad, nivel de detalle y simplicidad de implementación. Del artículo se toman las tecnologías que utiliza y el diseño en rasgos generales del backend.

Node.js, Express y PostgreSQL

PostgreSQL, también conocido simplemente como Postgres es un sistema gestor de bases de datos relacionales gratis y de código abierto. Como se describe en su documentación oficial (PostgreSQL, s.f.), es un manejador robusto que se encuentra disponible desde 1997. Está disponible en los sistemas operativos líderes: Linux, Windows, y macOS. Dado que PostgreSQL es conocido por su estabilidad, capacidad de extensión y cumplimiento de estándares, es una opción popular para desarrolladores y empresas. Se integra `node-postgres` (pg), que es un cliente PostgreSQL para Node.js que permite interactuar con la base de datos desde la aplicación.

Node.js (NodeJS, 2024) proporciona el entorno de ejecución para JavaScript en el servidor, mientras que Express es un framework web minimalista que facilita la creación de APIs. Express simplifica el manejo de rutas, *middleware* y respuestas HTTP.

Según la documentación oficial de Express (ExpressJS, s.f.), este es un *framework* minimalista, rápido y flexible para Node.js. Es uno de los más populares para esta tecnología y es muy flexible, por lo que se puede utilizar para abordar un amplio rango de tareas.

Sequelize

Sequelize, como se describe en su documentación (Sequelize, s.f.) es una biblioteca de Node.js que proporciona una herramienta de mapeo objeto-relacional (ORM) para bases de datos relacionales (en este caso PostgreSQL). Un ORM como Sequelize permite interactuar con la base de datos utilizando objetos y métodos de JavaScript en lugar de escribir consultas SQL directamente.

Además, la documentación oficial aclara que Sequelize proporciona una capa de abstracción sobre la base de datos, lo que significa que permite trabajar con modelos de datos en forma de objetos JavaScript en lugar de interactuar directamente con la base de datos utilizando SQL. Se pueden definir modelos de datos utilizando clases o definiciones de objetos JavaScript. Estos modelos se mapean a tablas en la base de datos, lo que facilita la creación, actualización, eliminación y consulta de datos. Además, simplifica la interacción con la base de datos, proporcionando métodos fáciles de usar para realizar operaciones CRUD

(Crear, Leer, Actualizar, Eliminar) y consultas complejas.

Integración con R

Mrgsolve es un paquete de R que permite correr modelos escritos en C++. Por lo tanto, en el backend se deben tener los modelos escritos en C++ y un ambiente donde se permita ejecutar *scripts* en R. Para poder ejecutar código en R dentro de Node.js, existe una biblioteca llamada *r-integration*. Como se explica en su documentación dentro del sitio de bibliotecas de NPM (R-Integration, 2022), esta permite combinar las ventajas de R con las de Node.js, con el fin de realizar cálculos avanzados en tiempo real y devolviéndolos al backend.

Se crean *scripts* en R que ejecutan los modelos, obtienen su información e interactúan con ellos. Luego, desde Node.js y mediante `r-integration`, se llaman y corren estos *scripts*.

5.1.1.2. Arquitectura de la API

Se crea una CRUD RESTful API en un ambiente Node.js que corre en un servidor Express y usa una base de datos PostgreSQL. Para conectar Express con PostgreSQL se utiliza `node-postgres`, permitiendo las operaciones CRUD (GET, POST, PUT, DELETE) en la API que va a correr los comandos de base de datos correspondientes.

Convenciones de rutas

Las rutas en la API están claramente definidas usando las convenciones RESTful y además siguiendo la división por entidad y componente. Para cada entidad poseemos un conjunto de rutas que comienzan con el nombre de la entidad y le sigue la definición según el objetivo de la misma. A modo de ejemplo, se tienen los siguientes *endpoints* para la entidad Paciente:

- GET `/patients`: Para obtener una lista de pacientes.
- GET `/patient/:ci`: Para obtener un paciente en específico.
- POST `/patient/:ci/add-manual-value`: Para agregarle un nuevo dato al paciente.

Utils

Se tienen en un directorio las funciones generales que se centralizan para su reutilización y evitar código duplicado. Existen diferentes archivos para: funciones relacionadas al manejo de errores, funciones genéricas pero que se reutilizan, funciones de validación y finalmente las constantes que definen las rutas de los *endpoints*.

Configuración

Los archivos de configuración se pueden encontrar en un directorio aparte, que consta de los archivos de configuración de la base de datos (el *seed* inicial,

la sincronización y configuraciones generales). Además se ubica un archivo de configuración de los roles y sus constantes.

Al crear una arquitectura como la ya descrita se apunta a facilitar los siguientes beneficios:

- **Escalabilidad**. El diseño modular hace que sea fácil agregar nuevas rutas, controladores, y modelos a medida que el sistema crece. Si se quiere añadir más funcionalidades (como nuevas entidades), solo se deben crear nuevos módulos, de forma que no afecte las partes ya existentes de la aplicación. Esto es ideal para proyectos que necesitan crecer con el tiempo.
- **Mantenibilidad**. La clara separación de responsabilidades (controladores para la lógica de negocio, servicios para la comunicación con la base de datos) genera código bien estructurado y fácil de entender. Esto reduce la complejidad cuando se necesita modificar el código o corregir errores. Si, p. ej., surge un problema en la base de datos, se puede corregir directamente en los servicios sin afectar la lógica de negocio o las rutas.
- **Validación**. El hecho de tener componentes bien definidos y aislados permite realizar pruebas unitarias de manera más eficiente. Cada parte del sistema puede probarse de forma independiente, lo que facilita la detección y corrección de errores. Además, como el acceso a la base de datos está abstraído en los servicios, es sencillo crear escenarios falsos para las pruebas. Esto permite simular la base de datos y verificar que los controladores y otras partes del sistema funcionen correctamente.

5.1.2. Eclipse Ditto

En esta sección se detallan conceptos importantes para el entendimiento de la plataforma Eclipse Ditto, así como también los puntos claves de su configuración y funcionamiento.

5.1.2.1. Conceptos importantes

Para comprender mejor el ecosistema Ditto, hay cierta información técnica y términos que es bueno conocer:

MQTT

MQTT (Message Queuing Telemetry Transport) es un protocolo de mensajería liviano diseñado para dispositivos con ancho de banda limitado o conexiones inestables. MQTT se usa mucho en el ámbito del IoT para la comunicación entre dispositivos y servidores. (MQTT, 2022).

Eclipse Ditto utiliza el protocolo MQTT para crear conexiones con los distintos dispositivos.

Eclipse Mosquitto

Eclipse Mosquitto[1] es un servidor de mensajes de código abierto que implementa las versiones 3.1 y 3.1.1 del protocolo MQTT. Mosquitto sigue el modelo de publicación/suscripción, donde los dispositivos pueden publicar mensajes en temas (topics) y suscribirse a temas para recibir mensajes. Mosquitto es un *broker*, es decir, es un intermediario que facilita la comunicación entre diferentes sistemas.

Policy

En el contexto de Eclipse Ditto, una *policy* (política) se refiere a un conjunto de reglas que se aplican a dispositivos, aplicaciones o usuarios dentro de un sistema IoT. Estas reglas definen los distintos permisos que tienen los usuarios sobre las distintas entidades.

Feature

En la plataforma Ditto, una Feature es una porción de información que es esperable que cambie. p. ej., la medida de un sensor puede ser modelada usando el concepto de Feature, ya que la información se va actualizando a medida que pasa el tiempo y las condiciones cambian.

Thing

En IoT, el término Thing refiere a cualquier dispositivo físico o virtual que tenga sensores, actuadores, y capacidad de conectividad para recopilar datos y enviar y recibir información.

En Eclipse Ditto, una Thing es una entidad genérica que se suele usar para agrupar varias features. p. ej., un sensor, un vehículo, una habitación u otros aspectos de la realidad pueden ser modelados usando el concepto de Thing.

5.1.2.2. Utilización de Ditto

En el marco de este proyecto de grado, se modela a cada paciente como una Thing, usando las features para representar los distintos tipos de datos de salud. P. ej., un paciente real podría ser la Thing identificada con su número de cédula y sus features podrían ser: nivel de glucosa en sangre, presión arterial y peso. A modo de ilustración, la Thing con ID **45678901** podría tener las siguientes features:

- Glucosa: 95 mg/dL
- Presión arterial: 120/80 mmHg
- Peso: 70 kg

Estos datos se actualizarían en tiempo real, cada vez que un sensor envíe información nueva acerca del paciente.

[1] https://mosquitto.org/

5.2. Presentación

En esta sección se detallan las decisiones tomadas con respecto a la implementación de las distintas aplicaciones que forman parte de la capa de presentación.

5.2.1. Panel de administración

Con el objetivo de que administradores del centro de salud puedan tener control sobre las entidades del sistema y realizar funciones importantes tales como crear pacientes, se crea una aplicación web utilizando AdminJS. Esta biblioteca permite crear un panel de administración de forma automática en base a una aplicación de Node.js, y brinda una interfaz gráfica que facilita realizar las operaciones GET, POST, PUT y DELETE sobre las entidades manejadas por ese *backend*. (AdminJS, 2024)

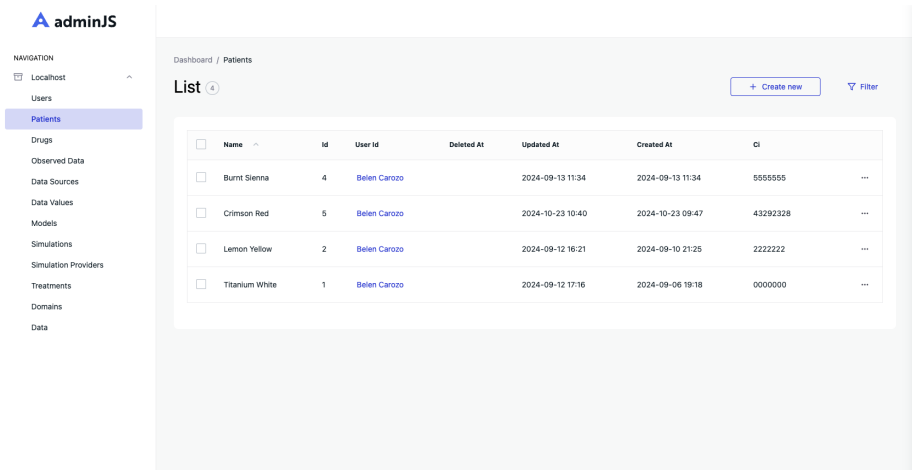


Figura 5.2: Panel de administración

5.2.2. Frontend

En esta sección se detalla la implementación de lo relacionado a la aplicación web, es decir, del *frontend* principal del proyecto.

5.2.2.1. Stack tecnológico

En esta subsección se mencionan las tecnologías usadas para la creación del *frontend* y sus respectivas justificaciones.

Framework: React.js con Typescript

Se decide usar React debido a la experiencia del equipo en el área, además de que es una de las tecnologías más usadas globalmente para proyectos frontend.

React, como se explica en su documentación oficial (ReactJS, s.f.) ofrece una arquitectura basada en componentes, un DOM (Document Object Model) virtual para actualizaciones eficientes del dibujado de las pantallas y una gran variedad de bibliotecas que se pueden utilizar para facilitar la programación. Existen otros *frameworks* que se podrían usar en este proyecto, p. ej. Angular. Por un lado, Angular se centra más en el buen rendimiento de aplicaciones más complejas, mientras que React optimiza su funcionamiento por lo mencionado del DOM virtual y la estructura de componentes.

Como se menciona en la documentación de React, este fue utilizado en plataformas globalmente conocidas como Facebook, Instagram, Whatsapp, Netflix, Twitter, AirBnb, y más, lo que muestra el soporte y confianza en el *framework* por desarrolladores de todo el mundo.

TypeScript es un lenguaje de programación fuertemente tipado que se basa en JavaScript y permite el uso de tipos dentro de la aplicación (Microsoft, 2024). Se prefiere sobre JavaScript porque proporciona un mejor control sobre el código, facilita la detección de errores y mejora la experiencia de desarrollo con características como autocompletado y documentación integrada.

Crear aplicación: Comparación de Vite con CRA

Existen varias herramientas que permiten la creación de proyectos React. Dos de las más usadas son Vite y CRA (Create React App).

En un artículo reciente de la industria se las compara (Vite vs Create-React-App, 2024) y se menciona que Vite requiere una mínima configuración comparado con CRA, lo que hace su configuración más rápida y fácil. CRA tiene rendimiento aceptable, pero Vite es conocido por su eficiencia y velocidad de compilación.

Además, el autor menciona que Vite ofrece gran flexibilidad en la personalización debido a su arquitectura basada en *plugins*. Vite es relativamente nuevo comparado con CRA, pero está siendo tendencia en las comunidades debido a su estabilidad y sus tiempos cortos de compilación. Por estas razones es que se decide usar Vite en lugar de CRA.

Enrutamiento: React Router

React Router proporciona enrutamiento declarativo, es decir, se le especifica qué pantalla se debe ver bajo qué ruta dentro de la aplicación. Esta estrategia de enrutamiento brinda la posibilidad de tener manejo del historial de navegación, y se puede realizar un cambio en la interfaz sin recargar toda la aplicación.

Además, tiene enrutamiento anidado, es decir, admite definir rutas secundarias dentro de las principales. Otra ventaja de esta estrategia es que permite definir y manejar parámetros por *url*. Debido a estas ventajas, hemos decidido utilizar React Router para el maneo de las rutas dentro de la aplicación. (React Router, s.f.)

Storage de datos: Comparación de Redux contra Zustand

En un artículo reciente de la industria (Redux vs Zustand, ¿cuál debo elegir?, 2023), se menciona que Redux es más sólido para aplicaciones grandes y complejas que centralizan el almacenamiento, mientras que Zustand está diseñada para aplicaciones más simples y está orientada a los *hooks*[2] de React.

Además, el autor menciona que Zustand utiliza un *hook* para acceder a los estados y para actualizarlos. Es ligero y performante debido a su minimalismo y uso eficiente de *hooks*. No es necesario escribir tanto código *boilerplate*, es decir, definiciones y configuraciones que deben ser siempre escritas, como en Redux. Como consecuencia se decide utilizar Zustand para el manejo de datos globales.

Manejo de conexión con API: RTK Query

RTK Query, según menciona su documentación oficial (RTK Query, 2024), junta tres funcionalidades en un sólo lugar: maneja estados, provee *hooks* para llamar a los *endpoints* y se encarga de las consultas HTTP.

Tiene un gran manejo de caché que evita tener que guardar estados en la mayoría de los casos. Además, posee *refetching*[3] automatizado mediante etiquetas (o *tags*) para invalidar entradas de la caché.

Una desventaja que tiene esta biblioteca es que lleva una gran cantidad de código *boilerplate*. Sin embargo, una vez configurado, agregar un *endpoint* es muy rápido y sencillo.

Para el manejo de respuestas, provee deserialización, manejo de errores y manejo de caso de éxito. También posee indicadores de tipo *boolean* del proceso de cada llamado y datos del estado de la consulta.

Calidad

Se establecen una serie de prácticas y herramientas fundamentales para garantizar la calidad y la mantenibilidad del código. Estas prácticas incluyen la implementación de herramientas como Eslint y Prettier, que permiten asegurar la consistencia y el cumplimiento de estándares de estilo en el código fuente. Adicionalmente, se usa TypeScript, que proporciona un sistema de tipado para JavaScript, incrementando la robustez del código.

Además, para facilitar la comprensión del código, se apunta a que los nombres de las variables sean claros e intuitivos. Se busca evitar el uso de código comentado como una solución temporal siempre que sea posible. La estructura de carpetas intenta seguir un esquema organizado que facilite la navegación y la escalabilidad del proyecto.

En busca de facilitar la mantenibilidad del código, se apunta a respetar convenciones de estilo y de nomenclatura recomendadas por TypeScript. La repetición de código busca ser evitada mediante la reutilización de componentes y funciones.

Diseño de interfaz gráfica web

Se decide utilizar una biblioteca de componentes y diseños base para tener

[2] Los *hooks* permiten utilizar más características de React sin depender de clases, manteniendo los componentes simples y funcionales (Meta, 2024).

[3] Consulta a la API para actualizar los datos.

un desarrollo centrado en la funcionalidad de la aplicación. En base al análisis realizado en la sección 5.5.1, se decide utilizar la biblioteca Ant Design. En base a los componentes de la biblioteca se crean componentes personalizados con diseños propios, para así poder reutilizarlos dentro de la aplicación.

Además, Ant Design provee una gran cantidad de componentes relacionados a la validación de formularios y manejo de eventos y estados. Lo más importante es que utilizando sus componentes, el proceso de crear y manejar formularios se integra fácilmente.

5.2.2.2. Consideraciones UI/UX

En esta subsección se menciona todo lo relacionado a las decisiones sobre la interfaz gráfica y experiencia de usuario.

Internacionalización

Se decide desarrollar la aplicación en inglés, ya que es el idioma más utilizado a nivel mundial. Dado que, según el análisis realizado en la sección 3.3, esta solución no tiene precedentes, podría ser adoptada en cualquier parte del mundo. A pesar de esta elección, se implementa una estructura que facilita la internacionalización de *strings*, permitiendo agregar traducciones de manera sencilla. Desarrollarla en inglés es una decisión estratégica que simplifica la traducción a otros idiomas, siempre que se cuente con alguien en el equipo que maneje este idioma. Si se quisiera agregar el idioma español, el procedimiento es sencillo: basta con utilizar las claves de cada *string* en inglés y crear un objeto de JavaScript equivalente para el español, aunque esto escapa al alcance de este proyecto.

Accesibilidad

Para que la aplicación pueda ser usada por una gran cantidad de usuarios, se pone foco en los estándares de accesibilidad de la industria. Esta es otra de las razones por las que usar una biblioteca de componentes es deseable, ya que estos componentes están construidos usando de manera adecuada los elementos de HTML semántico[4] (*header* como encabezado, *footer* como pie de página, entre otras cosas) y esto ayuda a las personas no videntes a poder desenvolverse en la página correctamente. Con el fin de verificar esto, se usa Lighthouse[5].

Se usa la semántica HTML de la forma que corresponde, para que la página pueda ser usada por mayor cantidad de tipos de usuarios. Un ejemplo de un mal uso de la semántica HTML es hacer que un elemento *div* sea presionable. Si un elemento es presionable, debe ser un botón u otro elemento detectable como tal, porque si no un lector de pantalla va a fallar al intentar comunicar la función de este elemento.

[4] Es el uso de las etiquetas HTML para reforzar el significado de la información en las páginas web, más que simplemente ser usadas para organizar visualmente la página.

[5] Lighthouse es una herramienta ofrecida por Google que permite auditar una página web con respecto a la accesibilidad, proporcionando un puntaje para cada una de las distintas pruebas que provee, así como también un puntaje promedio de estos para toda la página.

Se implementan los flujos de la aplicación teniendo en cuenta la navegación con teclado, buscando que todas las secciones de la página se puedan acceder sin necesidad de usar el *mouse*. Además, se buscar evitar que se resalten elementos que no sean útiles para el usuario. También es una buena opción evitar elementos presionables dentro de estructuras complejas como carruseles, porque estos dibujan todas sus páginas a la misma vez y la navegación con teclado se puede ver dificultada por esta razón.

Se revisan las etiquetas de accesibilidad asignadas a cada componente para asegurar que un lector de pantalla puede usarse para interpretar la página. Se evitan descripciones de elementos que no agregan información. Cada elemento debe dejar en claro su función al ser resaltado por un lector de pantalla.

5.2.2.3. Estructura

En esta sección se detalla la estructura que se tiene en cuenta al realizar el frontend. Se explica tanto a nivel de componentes como de estructura de directorios y archivos.

Componentes

Se crean componentes funcionales en lugar de clases de JavaScript, ya que estos poseen una sintaxis más concisa y clara que facilita la comprensión. A su vez, esta metodología permite reutilizar lógica mediante *hooks*.

Los *hooks* hacen que sea fácil extraer y reutilizar la lógica de estado en distintos componentes. También permiten escribir componentes funcionales más simples y ordenados, sin tener que lidiar con las clases ni con la palabra reservada *this* y sus complicaciones. Además, usando *hooks* se puede agrupar toda la lógica relacionada en un solo lugar, sin necesidad de dividirla entre diferentes métodos de ciclo de vida como ocurre con los componentes de clase.

Componente-Controller-Service

Se separan los componentes de los llamados a la API para diferenciar funcionalidades y clarificar código. De ser necesario, si el componente posee mucha lógica, se crea un controlador para ese componente en específico, lo que permite separar la lógica de la vista.

Estilos

Se definen estilos reutilizables en un archivo de estilos global. También se utilizan módulos de estilos, exclusivos a cada componente, en los casos en los que se considera necesario.

Directorios

Arquitectura de carpetas:

- /components
- /hooks
- /pages

- /stores
- /entities
- /layout
- /routing
- /assets

Modelo Humano

Debido a que la denominación «Gemelos Digitales» se trata de un concepto, queda como tarea a quien vaya a diseñar e implementar un sistema elegir la forma de representar al gemelo y los datos que lo acompañan de forma visual. Considerando la importancia otorgada a la usabilidad y diseño visual de esta aplicación, y en base a la prueba de concepto realizada en la sección 5.5.2, se decide implementar una representación gráfica del gemelo como la silueta de una persona.

Esta silueta cuenta los diferentes tipos de datos medidos por los *wearables* representados sobre el cuerpo. Se utilizan diseños creados manualmente para poder identificar al paciente y visualizar de forma sencilla y de fácil comprensión los datos del mismo y de dónde provienen. Para esto se dibujan los diferentes íconos de los datos de salud usando la herramienta Figma[6], en busca de tener un sistema de diseño consistente. También se dibuja una representación de las venas, para poder mostrar de forma gráfica y animada las distintas concentraciones disponibles de las simulaciones. En la figura 5.3 se puede ver el modelo humano descrito en esta sección.

[6] Aplicación que facilita la creación de diseños gráficos digitales.

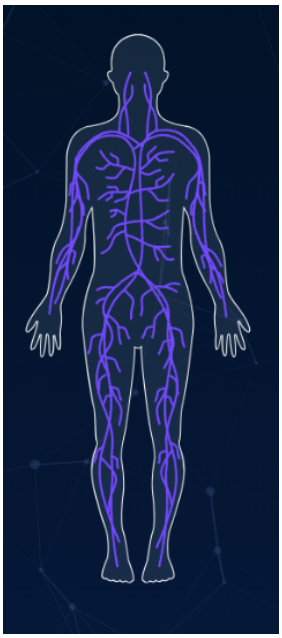


Figura 5.3: Modelo humano

5.2.3. Aplicación móvil

En esta sección se especifican las tecnologías utilizadas, la estructura definida para la aplicación móvil y otros aspectos importantes a resolver para cumplir con la funcionalidad esperada.

5.2.3.1. Tecnologías

Se toma la decisión de desarrollar una aplicación en React Native luego de investigar si la comunicación mediante MQTT y el envío de mensajes en *background* están soportados para esta tecnología. En esta decisión tuvo peso el hecho ya mencionado de que el equipo cuenta con experiencia de desarrollo en el *framework*, que es un marco híbrido que permitiría una extensión que se entiende sería directa a iOS en el futuro y que brinda soporte para las principales necesidades de desarrollo que son importantes para cumplir con los requerimientos de la aplicación.

5.2.3.2. Estructura

La estructura de esta aplicación es similar al proyecto React del frontend web de la plataforma. Cuenta con la carpeta raíz `src` que contiene:

- `assets`: Directorio en el que se almacenan todos los gráficos de la aplicación, ya sean imágenes, íconos, animaciones, entre otros.

- `componentes`: React Native usa React, por lo que se sigue el patrón de componentización recomendado por la librería.
- `hooks`: En esta carpeta se guarda código específico para realizar funciones que posee efectos secundarios, siguiendo nuevamente las recomendaciones de Meta para el desarrollo React.
- `sceens`: En este directorio se guarda el código relacionado a las pantallas de la aplicación, agrupado en subdirectorios por pantalla individual.
- `styles`: Se guardan los estilos generales de la aplicación, como p. ej. la paleta de colores.

5.2.3.3. Conexión con Health Connect

Health Connect (Google, 2024) es un servicio ofrecido por Google que permite centralizar los datos de salud del usuario en su propio dispositivo Android, y manejar qué aplicaciones tienen acceso a esos datos. Cualquier dato obtenido de un *wearable* que esté integrado con Health Connect podrá ser accedido por la aplicación móvil si el usuario otorga el permiso necesario.

Para que la aplicación React Native se comunique con Health Connect es necesaria la ejecución de código nativo que cumpla esta función. Esto se puede lograr implementando un módulo nativo o, si existe, usando una biblioteca que lo implemente.

Se decide usar la biblioteca `react-native-health-connect` de NPM en lugar de crear un módulo nativo, con el objetivo de facilitar y acelerar el proceso. Esta biblioteca implementa una interfaz que permite acceder a los datos de Health Connect directamente desde JavaScript. Además, provee una interfaz para pedir los permisos necesarios al usuario sobre los datos de salud, lo que apunta a brindarle al usuario total control sobre su información.

5.2.3.4. Conexión MQTT

Es necesario usar el protocolo MQTT para lograr la conexión con Eclipse Ditto. Para esto, existe la herramienta Paho: un proyecto de Eclipse que busca implementar el protocolo MQTT en Java para la plataforma Android.

Con el fin de aplicar esto a la solución, se utiliza la biblioteca `sp-react-native-mqtt`. Esta biblioteca proporciona un nivel de abstracción a la herramienta Paho para que la misma se pueda usar con React Native.

5.2.3.5. Background tasks

Con el fin de garantizar la comunicación de la aplicación en tiempo real y de evitar la necesidad de que el usuario tenga que abrir la aplicación para enviar actualizaciones de sus datos, se incorpora una estructura de tareas en background usando la biblioteca `react-native-background-actions`.

5.3. Proveedores de simulación

Un aspecto fundamental de este proyecto es la capacidad de realizar simulaciones utilizando todos los datos disponibles del paciente. Además, es esencial mantener la flexibilidad para no limitarse a una única simulación, contando con diversas opciones o, al menos, la posibilidad de configurarlas. Por ello, en esta sección se describen los distintos proveedores de simulación con los que cuenta MedFing y de qué forma los utiliza.

5.3.1. Finglix

El trabajo conjunto de los proyectos de grado de Drocco y cols. (2021) y Araujo y cols. (2024) produce una API que la aplicación descrita en este informe utiliza como proveedor de simulación. MedFing declara un formato general que deben cumplir todas las simulaciones, e implementa adaptadores que se encargan de transformar los datos de los distintos proveedores a ese formato universal.

De la API de Finglix se usan los *endpoints* relacionados con las simulaciones, es decir:

- `/auth/login/`. Este *endpoint* se utiliza para que el backend se autentique en la plataforma Finglix, utilizando un usuario destinado a la plataforma MedFing con el correo electrónico `medfingapi@gmail.com` y una contraseña que será guardada en las variables secretas de la aplicación Node.js.
- `/modeldrugs/`. Esta ruta se llama con la intención de obtener todos los modelos disponibles para simulación que provee Finglix.
- `/modeldrugs/:id/`. Este *endpoint* se usa para obtener la información específica de un modelo dado. Entre los datos brindados por esta llamada se encuentran los parámetros de simulación, que son usados para mostrar en el frontend como un formulario y poder ser ingresados por el usuario.
- `/modeldrugs/:id/simulate_dosis`. Con esta ruta se realiza una simulación con el modelo que se le pasa como parámetro de la mutación.

Todos estos *endpoints* son llamados desde el backend de MedFing. Esto se debe a que el mismo actúa como intermediario entre la aplicación y los servicios externos, gestionando la lógica de negocio y normalizando los datos antes de guardarlos y enviarlos al frontend.

Finglix permite dos tipos de simulaciones: poblacional e individual. En el marco de este proyecto, la integración con Finglix abarca solo el modelado poblacional. la razón de esto es que, para usar las simulaciones individuales es necesario que los pacientes de MedFing se creen también en Finglix, y que los datos ingresados en MedFing fluyan hacia Finglix. Esto se debe a que una restricción de Finglix es que para realizar simulaciones de tipo individual, el paciente debe tener al menos un dato cargado. Contar con esto conllevaría una integración más profunda entre MedFing y Finglix, y debido al alcance esperado

de este proyecto, todo esfuerzo relacionado con lograr dicha integración se deja como trabajo futuro.

5.3.2. Mrgsolve

Mrgsolve es un paquete de R para la simulación de modelos jerárquicos basados en ecuaciones diferenciales ordinarias (ODE), típicamente utilizados en el desarrollo de fármacos. Mrgsolve ha sido empleado en una amplia variedad de aplicaciones de modelos, incluyendo farmacocinética (PK), farmacocinética/-farmacodinámica (PK/PD), modelado farmacocinético basado en la fisiología (PBPK) y farmacología de sistemas cuantitativa (Mrgsolve, s.f.).

En la solución descrita en este documento, la herramienta se utiliza para brindar la posibilidad de integrar nuevos modelos que se vean beneficiados por las características de este proyecto de manera sencilla. Los profesionales encargados de generar los modelos ODE podrían representarlos usando el lenguaje de programación C++ y agregarlos al backend siguiendo las instrucciones presentes en el Anexo Configuración.

5.4. Despliegue

La aplicación MedFing basada en la solución propuesta en el capítulo anterior es una implementación compleja, que cuenta con diversos componentes que se deben desplegar en diferentes contenedores y sistemas. Es por esto que se dedica esta sección a explicar la arquitectura de despliegue de las distintas partes de la aplicación considerando las decisiones tecnológicas tomadas.

Considerando la existencia de leyes de protección de datos personales a nivel nacional, y dado que se manejan datos sensibles de pacientes, se opta por mantener estos datos dentro del país para que se rijan por estas leyes. Por esta razón se decide realizar el despliegue de las aplicaciones en servidores uruguayos y no en plataformas de nube como AWS y Heroku cuyos servidores se alojan en varias partes del mundo.

Al dejar desplegadas las soluciones, se agrega valor al proyecto permitiendo que se pueda acceder desde distintos sistemas. Sirve tanto para pruebas previas a que se utilice en el ámbito profesional como para el uso real de la aplicación. El despliegue consiste en un desafío importante ya que no se cuenta con experiencia en el equipo en el tipo de tareas involucradas a los despliegues, y tampoco con la plataforma de Antel. Esto trajo consigo varios problemas en el proceso, relacionados a la compatibilidad entre versiones de dependencias y bibliotecas de los proyectos.

5.4.1. Diagrama de Despliegue

Al diseño del diagrama de despliegue se le agregan los productos y conexiones correspondientes a las decisiones de implementación.

En la figura 5.4 se ilustra el diagrama de despliegue del sistema completo.

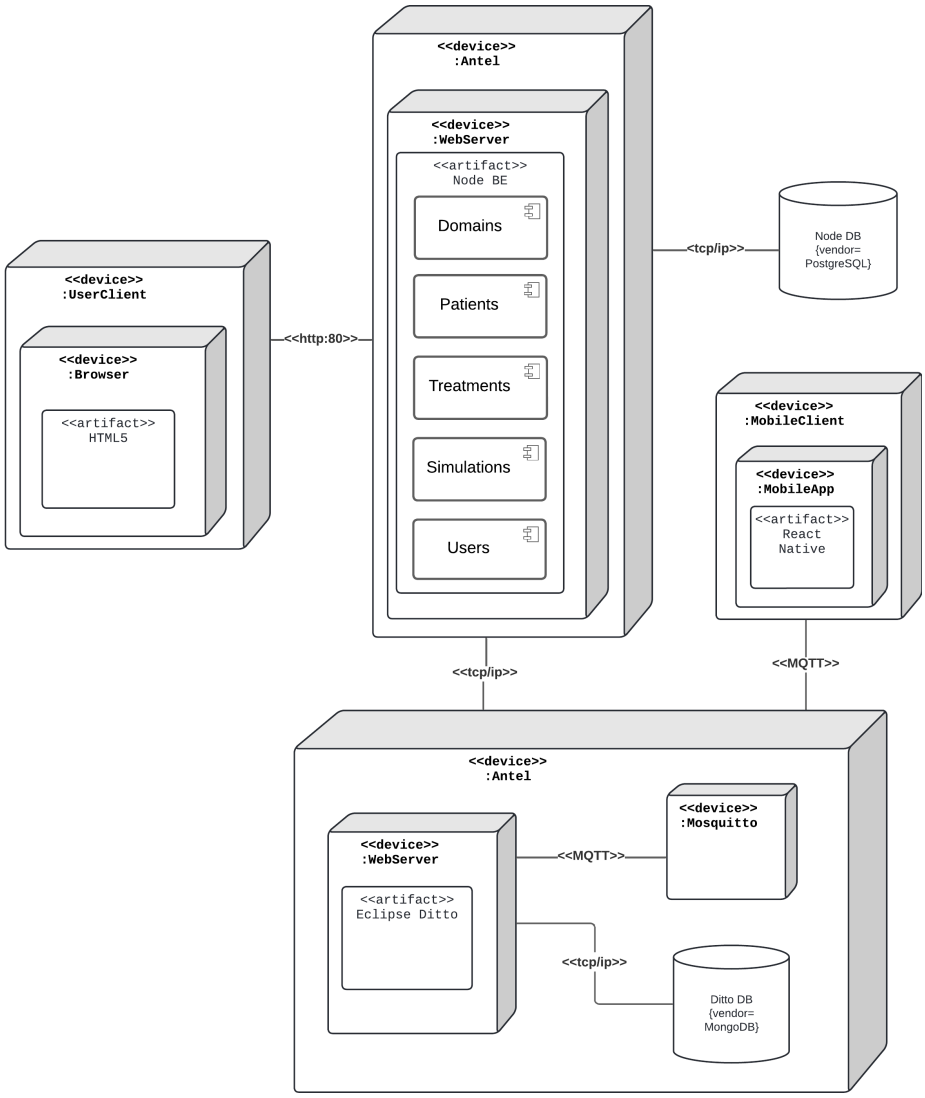


Figura 5.4: Diagrama de despliegue

5.4.2. Despliegue de Eclipse Ditto

Dado que ejecutar Eclipse Ditto requiere un uso significativo de recursos en la máquina *host*, se decide utilizar un código promocional de Mi Nube para desplegar un servidor de Ditto en Elastic Cloud. La estrategia inicial consiste en realizar el despliegue mediante contenedores de Docker, pero se encuentran diversas dificultades con este enfoque. A pesar de asignar más recursos a la

máquina en la nube, los contenedores no logran iniciarse debido a la falta de memoria disponible. Al revisar los registros del sistema, se identifica que el problema está relacionado con la memoria insuficiente, ya que Eclipse Ditto requiere una cantidad considerable de recursos, tales como dos núcleos de CPU y cuatro gigabytes de memoria dedicados.

Investigando más a fondo, se descubre que los desarrolladores de Ditto no recomiendan el uso de esta solución para entornos de producción, ya que fue diseñada para un entorno local. Esto lleva a explorar otras alternativas, como la recomendada por los desarrolladores de Ditto: Kubernetes.

El despliegue de una aplicación en Kubernetes requiere agregar el repositorio necesario, actualizarlo, y ajustar configuraciones. Entre estas, es clave limitar el uso de memoria para evitar problemas de rendimiento y establecer usuarios con permisos adecuados para acceso y administración.

Finalmente, para permitir el acceso externo a Eclipse Ditto, se configura un punto de entrada a través de la web, lo que permite verificar que el despliegue se realiza correctamente y que la aplicación está operativa. Además, se despliega un servidor adicional de Mosquitto para gestionar las conexiones MQTT. Este despliegue resulta complejo y costoso en tiempo debido a los múltiples inconvenientes que aparecen durante su realización. Se profundiza en los aspectos técnicos del despliegue de Ditto y obstáculos superados en el Anexo Configuración.

5.4.3. Despliegue del backend

El *backend* cuenta con diversos componentes y tecnologías, por lo que consiste en una pieza compleja de desplegar. Con el fin de facilitar el despliegue del *backend* se utiliza Docker. Esto permite agrupar la aplicación Node.js, R y sus dependencias, y la base de datos en un entorno aislado que pueda ser trasladado fácilmente a la nube, evitando problemas de compatibilidad causados por diferencias en sistemas operativos.

Usando Docker se orquestan los distintos servicios que se ofrecen, es decir, la base de datos y la aplicación Node.js. A su vez, se configura la imagen de la aplicación para que soporte el lenguaje R y los *plugins* necesarios para poder simular. La configuración de estas bibliotecas no es trivial, no falla inicialmente, pero posteriormente fallan los *scripts* de R porque no se encuentra el paquete `tidyverse`.

Aunque todos los paquetes parecen instalarse correctamente según los registros, al iniciar un contenedor se observa que el paquete `tidyverse` no está presente. Al intentar instalarlo de forma aislada, se identifica el problema: faltan algunas bibliotecas de Linux que no se especifican claramente como dependencias de `tidyverse`, como `libcurl4-openssl-dev`, `libssl-dev`, entre otras.

Luego de resolver este problema surge otro inconveniente: las simulaciones fallan sin razón aparente. Al analizarse la situación, se descubre que algunos números se procesan como *strings* en la imagen de Docker, mientras que en la aplicación ejecutada en la máquina *host* se tratan como números. Con esta información, se descubre que la versión de R instalada localmente no coincide

con la versión incluida en la imagen de Docker, lo que podría llevar a este tipo de inconsistencias. Los repositorios de Debian a veces están desactualizados, por lo que se decide descargar la versión más reciente de R desde CRAN y compilarla manualmente dentro de la imagen de Docker.

Al intentar utilizar la imagen de Node.js en Elastic Cloud, el despliegue falla debido a la falta de claves públicas para el gestor de paquetes `apt-get`. Es importante contar con este gestor, ya que se necesita instalar muchos paquetes que son dependencias de R. Las soluciones sugeridas en internet no resultan de mucha ayuda, y al investigar bastante se intuye que el problema radica en que la imagen de Node.js se genera con una versión más reciente de Docker que la disponible en Elastic Cloud, que es la versión 19. Se intenta utilizar Node.js 16, lo que permite que el gestor de paquetes funcione, pero muchas bibliotecas de nuestra aplicación fallan porque no soportan versiones de Node.js anteriores a la 18. Finalmente, se opta por instalar manualmente la versión más actualizada de Docker en la máquina de Elastic Cloud, resolviendo así el problema.

Finalmente, el último desafío del despliegue resulta ser comunicar Ditto con el backend. De manera inesperada, las consultas hechas desde la aplicación móvil generan un error fatal en el *backend* que impide que este continúe funcionando. Al investigar esto, se descubre que la biblioteca de Eclipse `@eclipse-ditto/ditto-javascript-client` se intenta conectar con una dirección IP de una red privada en lugar de la IP pública del entorno de Ditto. Se llega a la conclusión de que es la máquina de Elastic Cloud la que resuelve el dominio de Ditto a una red privada, probablemente de Elastic Cloud, pero a pesar de resolverlo a esa dirección IP no puede conectarse. Esto se resuelve finalmente editando el archivo `/etc/resolv.conf` y poniendo en primer lugar el DNS de Google (8.8.8.8) en lugar de los de Elastic Cloud. De esta forma, el dominio se resuelve correctamente.

5.4.4. Despliegue del frontend

Por la protección de datos, como se explica anteriormente, se decide utilizar un código promocional de Mi Nube para desplegar la aplicación web en Elastic Cloud de Antel y así evitar que los datos sean accesibles desde servidores en el extranjero.

Se despliega el *frontend* en un entorno con Ubuntu 22.04 (última versión disponible en Antel). Luego de crear el servidor correspondiente, se accede a él mediante la terminal del servidor proporcionada por la interfaz de Antel. Se configura una clave pública para agregar al repositorio y obtener permisos de acceso a él desde el servidor mediante *ssh*. Después, se instala todo lo necesario en la máquina creada, como *Git*, *Npm* y *Node*, ya que no vienen preinstalados. Con esto listo, se clona el repositorio y se instalan las dependencias necesarias del proyecto para generar un *build*. Previo a generar el *build* se crea un archivo en el *root* del proyecto llamado `.env` donde se establecen las variables de entorno. En este caso solamente se configura la variable de la URL del servidor de la API del *backend*.

El proyecto se ejecuta correctamente de forma local en el servidor, confirmando que la instalación es exitosa. Para finalizar el despliegue, se configura el acceso a la aplicación desde otras IP a través de la IP pública del servidor. Se decide utilizar Nginx (*NGINX*, 2023), un servidor web de código abierto que, entre otras funcionalidades, sirve código estático de manera rápida y eficiente. Esta elección se basa en su alto rendimiento y ligereza. Por lo tanto, se instala Nginx y se comienza su configuración.

La configuración requiere modificar los archivos generados durante la instalación. Se especifica la ruta donde se encuentra el *build* para desplegar. Para un uso más seguro y ordenado, se sugiere usar el directorio `/var/www/html` para guardar todas las aplicaciones estáticas por lo que se copian los archivos del *build* en el directorio mencionado y se ingresa en la configuración de Nginx. Se especifican además los puertos que deben escuchar en la IP pública (o dominio) para redirigir a esa aplicación estática. Se configuran los puertos HTTP por defecto (puerto 80) y se redirigen los accesos al código estático de la aplicación. Durante este proceso, surgen varios inconvenientes. Primero, es necesario configurar el *firewall* del servidor para que acepte cualquier acceso en los puertos indicados, ya que no funciona únicamente con la configuración estándar de Nginx. Solucionado este error, aún no se permite el acceso mediante la IP pública. Tras varias reconfiguraciones y una exhaustiva revisión de la documentación oficial de Nginx y Elastic Cloud (*Virtuozzo*, 2023), se identifica que la plataforma de Antel podría estar bloqueando cierto tráfico antes de que llegue al servidor. Esta suposición es correcta, ya que se encontró finalmente que existe un *firewall* externo al servidor además del propio *firewall* del servidor. Se procede entonces a habilitar los accesos correspondientes en él. Finalmente, se verifica que la aplicación queda correctamente desplegada.

5.4.5. Despliegue de la aplicación móvil

Dado que la aplicación móvil solo soporta Android en esta instancia, el despliegue de la misma consiste simplemente en generar un archivo APK (es decir, un programa para la plataforma Android) que los usuarios deben obtener e instalar siguiendo las instrucciones brindadas por el dispositivo. El mejor despliegue para los usuarios consistiría en publicar la aplicación en la Google Play Store, pero esto escapa el alcance de este proyecto de grado.

5.5. Decisiones tomadas

En esta sección se mencionan las pruebas de concepto realizadas en busca de tomar decisiones con respecto a la plataforma y validar hipótesis, de forma de mitigar riesgos en las etapas tempranas del proyecto.

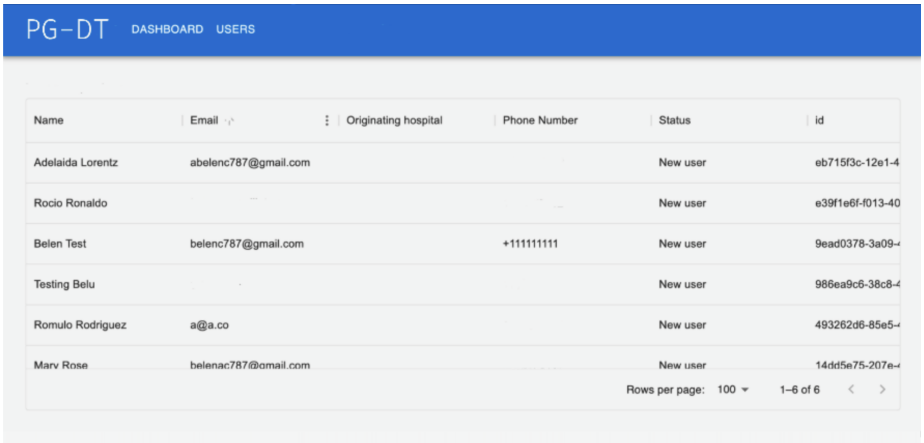


Figura 5.5: Prueba de concepto con Material UI

5.5.1. Frontend: Comparación entre Material UI y Ant Design

Para contar con las funcionalidades básicas de los componentes de interfaz gráfica, se decide usar una biblioteca de componentes. De esta manera, a la hora de implementar, es posible concentrarse en los requerimientos específicos de la plataforma y mantener un diseño cohesivo y útil sin tener que modificar componentes puros de HTML.

Se decide realizar una comparación entre Material UI y Ant Design, dos de las bibliotecas de componentes más usadas a nivel mundial. El objetivo es determinar cuál de las dos sería más adecuada en la práctica para este proyecto. Para ello, se crea una misma página utilizando ambas bibliotecas y se evaúa el proceso de desarrollo teniendo en cuenta diversos aspectos clave, como pueden ser: la apariencia visual, la calidad del código, el grado de personalización disponible, la calidad de la documentación y la accesibilidad ofrecida.

Material UI es una biblioteca de código abierto que usa el sistema de diseño de Google en los distintos componentes que ofrece. Por otra parte, Ant Design es una biblioteca desarrollada en China que ofrece una gran cantidad de componentes de alta calidad.

Con respecto a la **apariencia visual**, en la figura 5.5 se muestra una prueba simple realizada con MaterialUI usando componentes de *Header* y *Table*, mientras que en la figura 5.6 se realiza la misma prueba pero con Ant Design.

En el terreno de **código**, esta prueba concluye con que el código para esta pantalla usando Material UI alcanza un total de 153 líneas, lo que es considerado demasiado debido a la gran cantidad de opciones necesarias para la tabla. No hay una forma aparente de dibujado personalizado de celdas (hay otros componentes *Table* con un dibujado más personalizable pero a expensas de tener

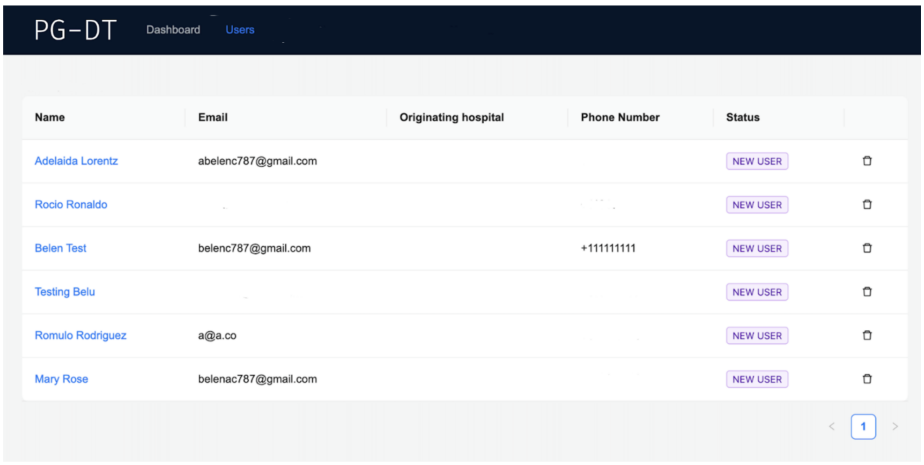


Figura 5.6: Prueba de concepto con Ant Design

que implementar otras características manualmente, como el filtro de orden). La tabla de Material UI ofrece soporte oficial para filtrado, orden, paginación, cantidad de filas por página y otras características. Crear esta pantalla sin experiencia previa, llevó aproximadamente aproximadamente una hora y treinta minutos. En caso de optar por la tabla simple, la que tiene celdas personalizables, sería necesario agregar código adicional para hacerla genérica. Esto se debe a que la disposición de las columnas no se proporciona en forma de arreglo, en su lugar, se hace con componentes. Los componentes son estáticos a menos que se implemente una iteración sobre los datos para dibujarlos.

Por otra parte, el código para esta pantalla llega a un total de 138 líneas usando Ant Design. En términos de líneas, se trata de menos código que Material UI, incluso teniendo en cuenta que el método para dibujar varias celdas es personalizado, a diferencia del caso de Material UI. Ant Design ofrece soporte oficial para filtrado, orden, paginación, cantidad de filas por página y otras características, al igual que Material UI. La pantalla fue rápida de crear, aproximadamente media hora sin experiencia previa. La configuración de las columnas se brinda a la tabla como un arreglo de objetos. Las únicas dos cosas necesarias para hacer una tabla simple se tratan de este arreglo y la información a mostrar en la tabla. Esto permite que la tabla sea dinámica, p. ej., haciendo posible cambiar el número de filas y columnas dependiendo de esas dos variables.

Con respecto a la **personalización**, la prueba de concepto no arroja resultados concluyentes acerca de si es posible personalizar la tabla usada de Material UI. Sin embargo, la tabla que es más simple es más personalizable, pero implica más trabajo para hacerla genérica por la forma en la que está construida. Por otra parte, el método de dibujado de las celdas de Ant Design es muy fácil de personalizar y la forma de hacerlo es muy intuitiva.

La **documentación** de ambas plataformas es buena. Material UI proporcio-

na muchos ejemplos en código que van al punto y no agregan cosas innecesarias, mientras que Ant Design proporciona código que no es necesario para el ejemplo y esto hace que la implementación no sea del todo clara en algunos casos.

En las pruebas realizadas con respecto a la **accesibilidad**, tanto en el caso de Material UI como de Ant Design, el lector de pantalla reconoce el *header* como un encabezado y la tabla como una tabla. No hay problemas de navegación con teclado y todos los botones son accesibles.

En el marco de este proyecto, se elige la biblioteca de componentes Ant Design, ya que es una biblioteca que ofrece una gran cantidad de componentes. Está muy bien documentada y es fácil de configurar y usar, además de que tiene un diseño moderno y estéticamente agradable. Es preferida sobre otras bibliotecas, principalmente por el alto nivel de personalización que ofrece, la calidad de la documentación, el gran soporte ofrecido por la comunidad y lo sencillo que es agregar componentes nuevos.

5.5.2. Frontend: Modelo humano

Para ofrecer una aplicación amigable y fácil de comprender, se decide utilizar algún elemento visual que permita observar al cuerpo del paciente y sus datos según de qué parte del cuerpo provengan. De esta forma se podrá visualizar específicamente de qué zona del paciente proviene el dato y se mejora la experiencia de usuario.

Se decide que la aplicación web tenga un paciente a cuerpo completo donde se puedan visualizar ciertos órganos o elementos del paciente de dónde se extraigan datos médicos y que al hacer *hover* con el *mouse* se pueda ver el último dato del paciente relacionado a esa parte del cuerpo y la fecha en el que fue tomado. A su vez, se decide que para mediciones relacionadas con las concentraciones en sangre de cierta droga se podrá observar un rango de colores imitando la intensidad de la concentración. Es decir, si la concentración en sangre es baja, las venas tendrán un color claro y si la concentración es alta se tendrá un color más fuerte. A mayor concentración, mayor opacidad.

Se consideran dos opciones para realizar el modelo del paciente. La primer opción es investigar cómo subir un modelo *3D* a React y que se pueda interactuar con él como girarlo, seleccionar órganos y mostrar datos. La otra opción es crear un *png* de la silueta del paciente manualmente y a partir de ahí agregar íconos u otros *png* para especificar órgano, venas y lo que sea necesario.

La biblioteca encontrada para utilizar modelos *3D* en React es **Three.js**. Esta biblioteca de JavaScript, como se explica en su documentación oficial (Three.JS, s.f.), permite crear y mostrar gráficos *3D* en el navegador web. Para poder utilizarla con React, se debe también utilizar otra biblioteca a modo de conexión. El proceso de trabajar con modelos *3D* se explica en su documentación y es detallado a continuación. Primero se deben instalar todas las dependencias y bibliotecas necesarias. Luego, se debe tener un modelo *3D* exportado en *GLTF* o *GLB* y que estén optimizados para la web. Este modelo se debe colocar en la carpeta *public* del proyecto y luego para dibujarlo en la aplicación se deben utilizar componentes específicos de la biblioteca de conexión donde se pueden

editar todos los estilos del modelo ya sean elementos de la escena, de cámara, luces y controles como rotación, *zoom*, etc. Esto permitiría un manejo del modelo simple y útil brindando movimientos para facilitar al médico analizarlo. En cuanto a usabilidad y gráficos es la opción ideal, pero se necesitaría dedicación de tiempo para crear todo el modelo *3D* del paciente y de los órganos, venas y detalles, ya que los modelos existentes son pagos.

La otra opción mencionada es crear un *png* manualmente dibujando la silueta de un paciente. Esto sería simple ya que es la silueta del paciente y las venas. Se usarían iconos para indicar lugares específicos u órganos del paciente y luego con Ant Design se crearían componentes para visualizar los datos como *pop-ups* al hacer *hover* con el cursor. Para el color de las venas se dibujarán las venas en *png* y luego se le modificaría el estilo y la paleta de colores según el color que se requiera. Es una forma sencilla pero no tiene tantas funcionalidades como utilizar un modelo *3D* ya que un *png* es completamente estático.

Como decisión final, se concluye que crear el modelo manual es la mejor opción por varias razones. Lo principal es el tiempo a invertir en esta funcionalidad del proyecto. Es una funcionalidad más visual que lógica por lo que no es lo esencial de la aplicación, entonces se decide que no se debería invertir demasiado tiempo en ello teniendo otras prioridades. El modelo *3D* requeriría investigación en cómo utilizar, p. ej., Blender, que es una aplicación para crear modelos y que no es sencilla de aprender. Además, estéticamente, tener la rotación y poder mover el paciente agrega una mejora visual pero no aporta utilidad real, por lo tanto un modelo estático bastará para cubrir las necesidades. Otra ventaja es que al crear un modelo manual se le pueden agregar los diseños y detalles que se quiera al mismo y tiene total posibilidad de personalización.

En conclusión, se utiliza un modelo estático manual para poder representar los datos del paciente y se integra con Ant Design para una mayor personalización.

5.5.3. Gemelos Digitales: Eclipse Ditto y la conexión con dispositivos wearables

El punto más fuerte de la comparación entre plataformas, y el que finalmente hace pesar más en la decisión a Eclipse frente a las otras, es que se trata de una plataforma gratuita y de código abierto. Es por esta razón que finalmente se opta por integrar Eclipse Ditto como proveedor de infraestructura de Gemelos Digitales para la plataforma de precisión de dosificación.

Para corroborar la validez de Eclipse Ditto como plataforma útil para el caso de uso descrito en este documento, se realiza una prueba de concepto con el objetivo de determinar si es posible crear una conexión exitosa entre un *wearable* y Eclipse Ditto. En el Anexo Configuración se explica la prueba con mayor detalle técnico.

La prueba consiste en clonar el repositorio de Eclipse Ditto, configurar la autorización con una clave de administrador e iniciar la plataforma localmente. También se debe clonar un ejemplo de código abierto de simulador de *iWatch*, que va a oficiar de *wearable* en esta prueba.

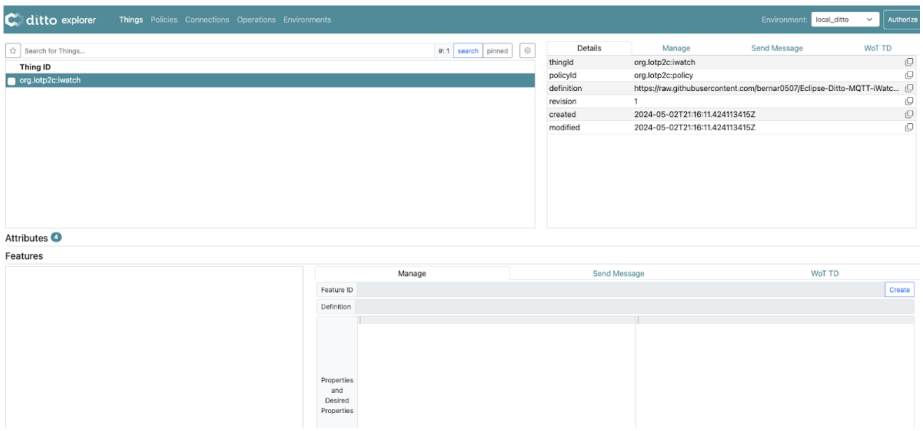


Figura 5.7: Verificación de la creación de la *thing*

Además, se deben realizar configuraciones para generar una conexión que oficie de puente entre Ditto y el simulador de *iWatch*.

El siguiente paso es crear la política de acceso a los datos del *wearable*, conectándose con Ditto mediante HTTP y usando el usuario de administrador previamente configurado para hacer modificaciones en el sistema. De esta forma se definen las reglas sobre los recursos. Se puede corroborar la creación de la política de acceso en la interfaz gráfica que expone Ditto.

Con la misma metodología de comunicación con Ditto mediante HTTP, se define el esquema de datos que representa al *wearable* y se la crea asignándole un nombre y la política de acceso creada en el paso anterior. También se puede verificar la creación de este esquema de datos mediante la interfaz gráfica que ofrece Ditto (figura 5.7)

Finalmente, se instalan las dependencias necesarias y se ejecutan los archivos que corresponden al simulador de *iWatch*. Este *script* se encarga de enviar mensajes al tópico correspondiente para imitar el comportamiento de un dispositivo conectado a la red *IoT*. Para corroborar que la conexión funciona de la forma esperada, se ejecuta una *request* GET a Eclipse Ditto, usando como *path* el tópico en el que los mensajes fueron publicados. Si los mensajes se muestran de forma correcta, se puede concluir que la conexión fue exitosa, como se puede ver en la figura 5.8.

5.5.4. Obtención de datos en tiempo real del paciente

En esta sección se investigan las posibles formas de recabar datos usando *wearables* y se explican las consideraciones tomadas para elegir implementar una aplicación móvil como interfaz en lugar de desarrollar software para un *wearable* en particular. Se detallan las distintas opciones consideradas para lograr

Figura 5.8: Verificación de la conexión

comunicar los dispositivos *wearables* con Eclipse Ditto.

5.5.4.1. API de Mi Fit (Xiaomi)

Huami, fabricante de *wearables* Xiaomi, brinda acceso a una API que permite acceder a los datos registrados por estos *wearables* (ZeppHealth, 2020). Sin embargo, hay que seguir un proceso bastante largo para registrar la aplicación y tener acceso a los datos. Es necesario proporcionar una imagen como logo, brindar dirección del sitio web, entre otros requisitos. Además, solo se permite registrar usuarios corporativos. Es por esto que, de decidir usar la API, se debería registrar la aplicación a nombre de la Facultad.

Algunos de los datos que se pueden obtener usando esta API son: sexo, altura, peso, pasos, consumo de calorías, datos del sueño y ritmo cardíaco.

Una desventaja de este enfoque, además del largo proceso de registro de aplicaciones, es que solo se permite acceder a los datos brindados por dispositivos fabricados por Huami.

5.5.4.2. Relojes inteligentes de Apple y Google

Una solución considerada se trata de desarrollar software específico para watchOS y WearOS (equivalentes a iOS y Android para relojes inteligentes, respectivamente) que se encargue de registrar los datos de salud del paciente y se conecte con Eclipse Ditto.

Esto tiene varias desventajas. En primer lugar, no existen tecnologías híbridas para desarrollar software en relojes inteligentes, por lo que las aplicaciones para watchOS y WearOS no compartirían código y deberían ser desarrolladas en sus respectivos lenguajes nativos. Por cuestiones de alcance, y por la falta de experiencia del equipo en desarrollo móvil nativo, tener que desarrollar estas aplicaciones reduciría de forma significativa el tiempo disponible para dedicarle a otras partes de la plataforma.

En segundo lugar, es imprescindible que el reloj tenga conexión a internet. Sin esto, la solución sería inútil puesto que no podría comunicarse con Ditto. No todos los *wearables* tienen acceso a internet, muchos se comunican solo con el teléfono celular mediante bluetooth, lo que reduciría el rango de dispositivos en los que la aplicación podría funcionar.

Finalmente, los *wearables* tienen un rango acotado de datos de salud que pueden medir. De elegir este enfoque, la plataforma perdería la posibilidad de acceder a datos médicos potencialmente valiosos para el tratamiento.

5.5.4.3. Simulador de smartwatch

Otra opción evaluada se trata de usar el simulador de MQTT en Python que fue integrado con Eclipse Ditto en la prueba de concepto, y dejar como trabajo futuro la integración de otros *wearables*, teniendo ya la infraestructura para hacerlo. Usar esto en lugar de un dispositivo real es menos conveniente en términos de completitud de la solución, pero, de igual manera, la opción fue tomada en consideración por una cuestión de alcance.

5.5.4.4. HealthKit y Health Connect

HealthKit (Apple, 2024) es un servicio ofrecido por Apple que recopila todos los datos de salud de un usuario. Estos datos se guardan localmente en el celular de cada usuario, por lo que no existe una API web o similar para obtenerlos. La única forma de obtener estos datos sería crear una aplicación móvil que se comunique con Ditto.

Esta herramienta ofrece una gran cantidad de datos, como p. ej.: porcentaje de grasa corporal, glucosa en sangre, frecuencia cardíaca, saturación de oxígeno, presión arterial, y muchos más.

Health Connect (Google, 2024) es un servicio muy similar a HealthKit, pero para Android. Health Connect reemplaza a un servicio previo de Google llamado Google Fit, una API web que fue deprecada. La diferencia entre Google Fit y Health Connect radica principalmente en que, mientras que Google Fit guardaba los datos en la nube, Health Connect los almacena localmente en el dispositivo móvil del usuario. Esto proporciona beneficios en términos de privacidad de datos y seguridad. Al igual que con HealthKit, la única forma de obtener los datos de salud del paciente sería crear una aplicación móvil que se comunique con Ditto.

Esta herramienta provee datos de glucosa en sangre, presión arterial, grasa corporal, temperatura corporal, masa de agua corporal, masa ósea, frecuencia cardíaca, variabilidad de la frecuencia cardíaca, saturación de oxígeno, entre otros.

HealthKit y Health Connect centralizan todos los datos de salud. Es decir, en ellas se pueden almacenar tanto datos de los pasos dados en el día como datos de valores de glucosa en sangre, que sean brindados por un *wearable* específico, manejado por una aplicación específica. Todas las aplicaciones de salud se comunican con estas herramientas y, teniendo los permisos necesarios, todas las aplicaciones pueden acceder a todos los datos. Esto es sumamente útil, porque permite prescindir de integrar un *wearable* específico para pasar a soportar todos los *wearables* que tengan integración con estos servicios.

Una aplicación WearOS o watchOS puede conectarse directamente con Eclipse Ditto usando el protocolo MQTT. Esta solución es lo más parecido a lo que

inicialmente se visualizaba. Sin embargo, esto limita el alcance a los sensores de ese reloj inteligente en específico, a diferencia de Health Connect. Además, esta solución se tiene que desarollar en Android nativo, una tecnología en la que el equipo no tiene experiencia. Otra ventaja de usar HealthKit y Health Connect es que se puede desarrollar la aplicación usando React Native, un marco de desarrollo móvil híbrido que permite compartir código iOS y Android, y en el que además el equipo cuenta con experiencia.

En base a las ventajas planteadas de esta alternativa, se decide seguir adelante con ella usando Health Connect y desarrollando para Android, pero creando la infraestructura para que se pueda extender fácilmente a HealthKit en iOS.

Capítulo 6

Pruebas y Validaciones

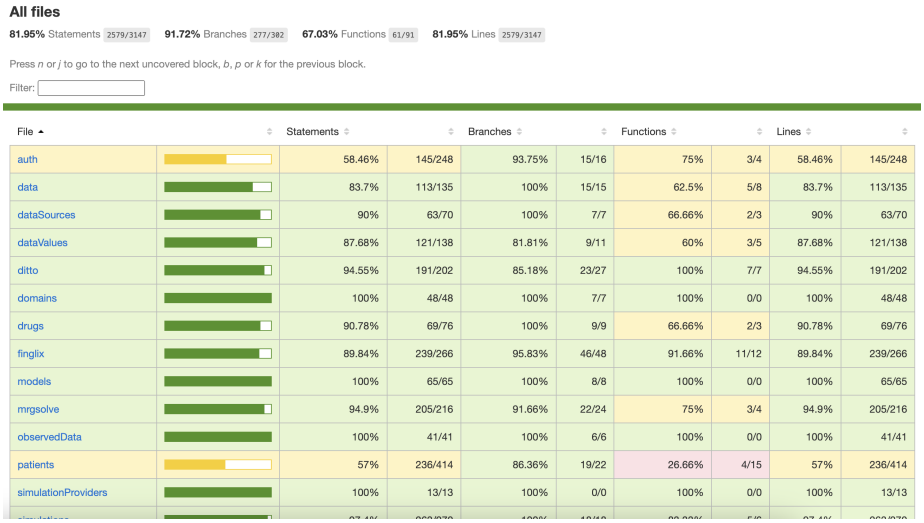
En este capítulo se presentan las pruebas realizadas para evaluar el correcto funcionamiento y la calidad del producto. Se describen los distintos tipos de pruebas efectuadas y los criterios utilizados para su diseño, así como los resultados obtenidos, con el objetivo de evaluar el cumplimiento de los requisitos establecidos.

6.1. Pruebas del Sistema

En esta sección se explican las pruebas aplicadas al sistema completo, evaluando tanto su comportamiento general como la interacción entre los distintos componentes. Estas pruebas tienen como objetivo identificar posibles fallos de integración, revisar los distintos flujos y verificar la calidad del código que compone al sistema.

6.1.1. Pruebas unitarias sobre el backend

Se decide realizar pruebas unitarias sobre el *backend*, parte central de la aplicación ya que maneja las simulaciones, proveedores, pacientes y es el intermediario con los Gemelos Digitales. Se crean pruebas con el fin de verificar el funcionamiento de los flujos críticos del sistema y de prevenir errores al agregar código en el futuro. Para esto se utiliza la herramienta Vitest, una libraría de Vite que brinda la posibilidad de crear y ejecutar pruebas unitarias de manera sencilla y rápida. Se crean 90 pruebas unitarias para evaluar la aplicación *backend*. Vitest también permite realizar un reporte de *coverage*, que llega a alcanzar un 80 % de líneas cubiertas para los servicios, modelos y controladores. En la figura 6.1 se puede ver la interfaz gráfica generada por Vitest que brinda información acerca del *coverage* total, así como también el *coverage* específico por módulo.



All files

81.95% Statements 2579/3147 **91.72%** Branches 277/302 **67.03%** Functions 61/91 **81.95%** Lines 2579/3147

Press *n* or *j* to go to the next uncovered block, *b*, *p* or *k* for the previous block.

Filter:

File	Statements		Branches		Functions		Lines	
auth	58.46%	145/248	93.75%	15/16	75%	3/4	58.46%	145/248
data	83.7%	113/135	100%	15/15	62.5%	5/8	83.7%	113/135
dataSources	90%	63/70	100%	7/7	66.66%	2/3	90%	63/70
dataValues	87.68%	121/138	81.81%	9/11	60%	3/5	87.68%	121/138
ditto	94.55%	191/202	85.18%	23/27	100%	7/7	94.55%	191/202
domains	100%	48/48	100%	7/7	100%	0/0	100%	48/48
drugs	90.78%	69/76	100%	9/9	66.66%	2/3	90.78%	69/76
finglix	89.84%	239/266	95.83%	46/48	91.66%	11/12	89.84%	239/266
models	100%	65/65	100%	8/8	100%	0/0	100%	65/65
mrgsolve	94.9%	205/216	91.66%	22/24	75%	3/4	94.9%	205/216
observedData	100%	41/41	100%	6/6	100%	0/0	100%	41/41
patients	57%	236/414	86.36%	19/22	26.66%	4/15	57%	236/414
simulationProviders	100%	13/13	100%	0/0	100%	0/0	100%	13/13

Figura 6.1: Coverage de código del backend

6.1.2. Análisis estático de código

Con el fin de mantener la calidad, legibilidad y prolijidad del código, se utiliza la herramienta Eslint para realizar un análisis estático de código para verificar que se cumplan ciertas buenas prácticas de legibilidad estándar de JavaScript y React. Se ejecutan las pruebas sobre la versión final del código tanto en la aplicación frontend web como en la aplicación móvil. Los resultados obtenidos se pueden ver en las figuras 6.2 y 6.3, respectivamente. En ninguna de las bases de código hay errores, solo existe una *warning* menor en el caso de la aplicación web.

```
❯ npx eslint .

/Users/belencarozo/projects/medfing-app/medfing-ui/src/components/lineChart/LineChart.compo
nent.tsx
  14:14  warning  Fast refresh only works when a file only exports components. Use a new fi
le to share constants or functions between components  react-refresh/only-export-components

✖ 1 problem (0 errors, 1 warning)
```

Figura 6.2: Revisión de la aplicación web usando Eslint

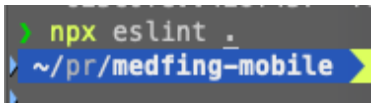


Figura 6.3: Revisión de la aplicación móvil usando Eslint

También se realizan las pruebas de TypeScript para corroborar que no haya ofensas[1] en el código, usando la herramienta `tsc`. Los resultados de estas pruebas para la aplicación web se pueden ver en la figura 6.4, y en la 6.5 para la aplicación móvil. En ambos casos, no se registra ninguna ofensa.

Figura 6.4: Revisión de la aplicación web usando tsc

Figura 6.5: Revisión de la aplicación móvil usando tsc

6.1.3. Métricas de la web

Como fue mencionado anteriormente, para obtener las métricas de la web se utiliza la herramienta Lighthouse. Los parámetros a analizar por esta herramienta son: rendimiento, accesibilidad y buenas prácticas.

Se analiza la página principal del paciente, y los resultados son los de la figura 6.6. Como se puede ver, las métricas tienen resultados excelentes, tanto en términos de accesibilidad como de buenas prácticas. El área con más posibilidad de mejora se trata del rendimiento, con un porcentaje de 52 %. Las medidas que se toman para llegar a este resultado consideran, p. ej., cuánto se demora en dibujar un componente con contenido en la pantalla. Una posible razón de este resultado de rendimiento puede ser la forma en la que se obtienen los pacientes por diseño; se opta por traer los datos de Eclipse Ditto y entregarlos junto con el resto de los datos del paciente. Una posible mejora podría ser separar estas llamadas, para que se puedan dibujar componentes en la pantalla que no dependan de estos datos y así mejorar las métricas. Considerando que el rendimiento se encuentra en la franja intermedia (no es inaceptable pero tampoco es excelente), se toma la decisión de que estas mejoras entren dentro del trabajo a futuro.

También se analiza otra página importante para la aplicación: la página de los tratamientos y simulaciones (figura 6.7). Los resultados son muy favorables con respecto a accesibilidad y buenas prácticas, mientras que el test de rendimiento no arroja resultados concluyentes.

[1] Errores de tipos o de estilo no fatales, pero que no cumplen con las buenas prácticas recomendadas por TypeScript.

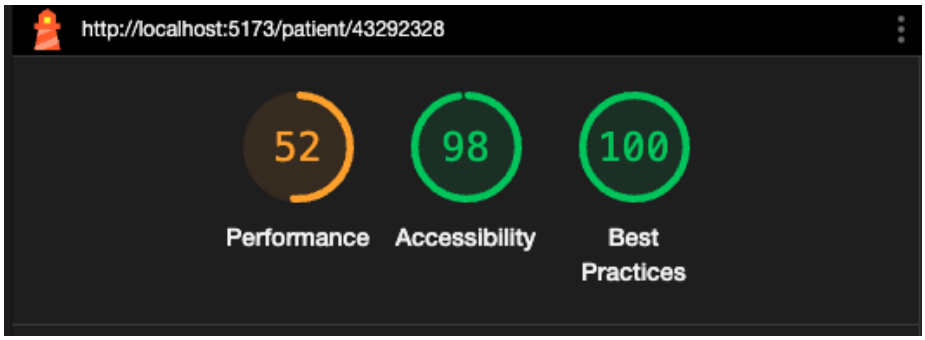


Figura 6.6: Reporte de Lighthouse para la página de visualización de paciente

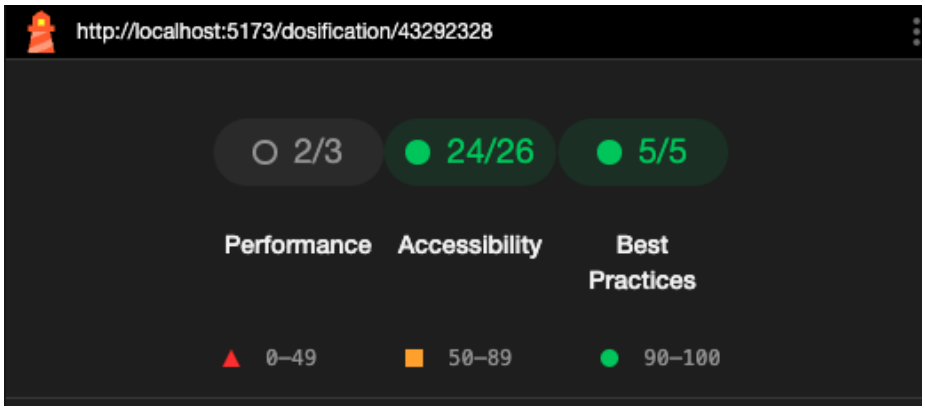


Figura 6.7: Reporte de Lighthouse para la página de tratamientos

6.1.4. Lectores de pantalla

Los lectores de pantalla son herramientas de asistencia tecnológica que permiten a personas con discapacidad visual interactuar con interfaces digitales a través de descripciones auditivas del contenido en pantalla. En este proyecto, se utilizan los lectores de pantalla VoiceOver[2] en MacOS y TalkBack[3] en Android para evaluar la accesibilidad de las aplicaciones.

Para corroborar que las aplicaciones son accesibles con lectores de pantalla, se recorren todos los flujos usando exclusivamente esta forma de navegación. El resultado de esta prueba es que todos los estados de la aplicación son alcanzables usando un lector, a pesar de que existe área de mejora con respecto a textos alternativos poco descriptivos, elementos que se seleccionan varias veces en el

[2]https://support.apple.com/guide/voiceover/welcome/mac
[3]https://support.google.com/accessibility/android/answer/6283677?hl=en

mismo ciclo y otras características. Para evaluar correctamente las mejoras necesarias, se debería priorizar y dedicar tiempo a una auditoría de accesibilidad. Dentro del contexto y alcance de este proyecto, se considera suficiente que los flujos sean alcanzables, que los roles de los elementos estén bien definidos y que existan textos alternativos para las imágenes.

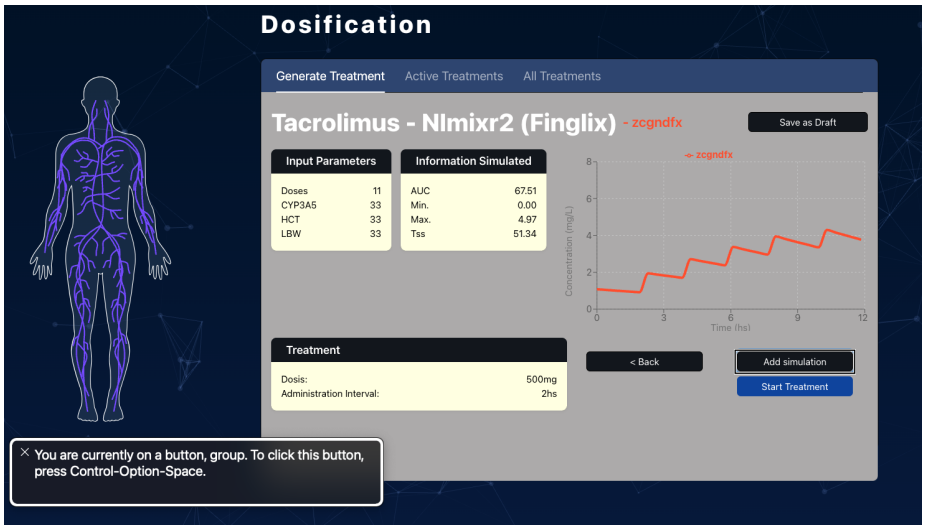


Figura 6.8: Revisión de accesibilidad usando Voice Over

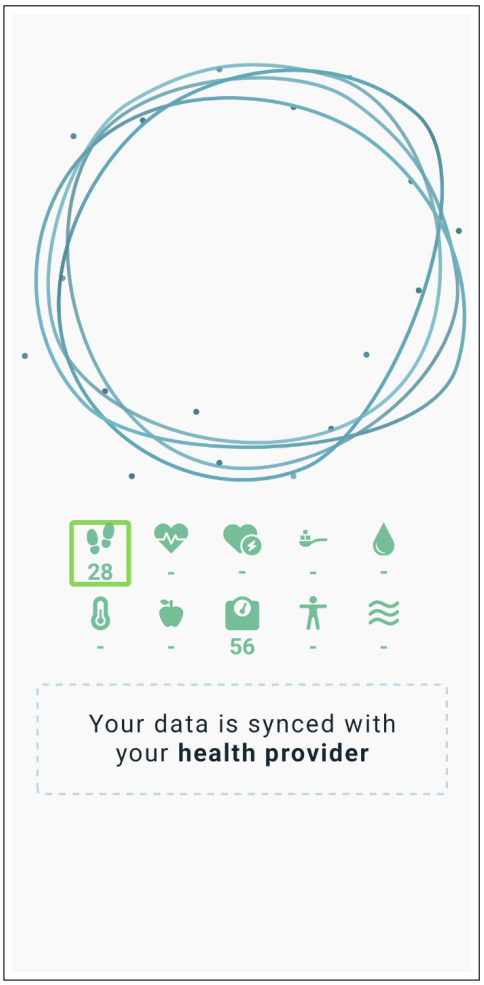


Figura 6.9: Revisión de accesibilidad usando Talk Back

6.1.5. Pruebas de flujo

En base a los requerimientos, se construyen los flujos necesarios usando la infraestructura provista por la combinación de implementación de la propuesta del capítulo de diseño y las herramientas y consideraciones mencionadas en el capítulo sobre implementación. Se realizan pruebas manuales sobre cada uno de estos flujos para corroborar la calidad de la solución y el correcto funcionamiento de los mismos. En el Anexo Manuales de usuario se encuentra más información acerca de los distintos flujos probados.

6.2. Validación con potenciales clientes

En esta sección se describen las estrategias utilizadas para recopilar comentarios, sugerencias y validar la propuesta con potenciales clientes.

6.2.1. Reuniones con cliente

Se tienen reuniones con el investigador Manuel Ibarra, del grupo GIDP, que oficia como «cliente» del producto de esta tesis. En estas reuniones se muestra el producto en sus distintas etapas, con el objetivo de obtener validación de la solución por parte de un investigador dedicado al área de dosificación de precisión. Estas reuniones derivan en los siguientes aspectos prácticos aplicados al proyecto:

- **Integrar un proveedor de simulación que permita simular usando modelos PB/PK.** Esta sugerencia nace en base a que este tipo de modelos requieren una gran cantidad de datos, una característica ideal para un enfoque basado en Gemelos Digitales como es el de este proyecto. Este consejo desemboca en la integración de Mrgsolve como proveedor de simulación, que no solo permite integrar modelos PB/PK sino que también permite cualquier tipo de modelo ODE.
- **Agregar tipos de datos para monitorear usando Eclipse Ditto.** En base a la documentación de Health Connect y los tipos de datos de salud que provee, se sugiere una lista de diez tipos de datos a integrar en el proyecto: `BloodPressure`, `HeartRate`, `BloodGlucose`, `RestingHeartRate`, `BasalBodyTemperature`, `BasalMetabolicRate`, `Weight` y `LeanBodyMass`. Esto produce la posibilidad de monitoreo de los mismos en la aplicación móvil y en el modelo humano de la aplicación web.

6.2.2. Seminario LINS

Se lleva a cabo una instancia de validación con investigadores y estudiantes de posgrado del grupo GIDP e integrantes del LINS[4], con el fin de evaluar la solución desarollada y discutir distintas mejoras, sugerencias y aportes a futuro propuestos por los participantes.

Para esta instancia se crea una presentación que se muestra a los participantes, buscando instruirlos en los conceptos de dosificación de precisión, Finglix y Gemelos Digitales para que comprendan mejor el objetivo de la plataforma.

Posteriormente, se realiza una demostración de las principales funcionalidades de la plataforma en tiempo real. También se realiza una ronda de preguntas y comentarios, con el fin de que los participantes puedan expresar su opinión de forma oral. En esta ronda se hacen buenas preguntas y aportes muy valiosos.

Finalmente, se pide a los participantes que completen una encuesta luego de la sesión de presentación. Las encuestas se diseñan mediante Google Forms y

[4] https://www.fing.edu.uy/es/inco/laboratorio-de-integraci %C3 %B3n-de-sistemas

tienen como objetivo recopilar sus percepciones y sugerencias sobre el sistema. A continuación se detallan las preguntas de la encuesta, agrupadas en distintas secciones.

6.2.2.1. Contexto

1. ¿Conocía previamente el concepto de gemelos digitales? **Sí / No**

2. En su opinión, ¿cuán importante considera la personalización en los tratamientos médicos? **(del 1 al 5, donde 1 es "Nada importante" y 5 es "Extremadamente importante")**

3. ¿Cree que los gemelos digitales podrían mejorar la dosificación de precisión? **Sí / No / No estoy seguro(a)**

4. ¿Ha utilizado alguna vez herramientas para simulaciones de dosis de medicamentos? **Sí, regularmente / Sí, en ocasiones / No**

6.2.2.2. Objetivos

1. ¿Qué tan útil cree que es una plataforma que combine simulaciones avanzadas y datos en tiempo real de pacientes? **(del 1 al 5, donde 1 es "Nada útil" y 5 es "Extremadamente útil")**

2. Según su experiencia y opinión, ¿qué características considera esenciales en una herramienta de dosificación de precisión? **(Respuesta libre)**

3. ¿Qué limitaciones o problemas identifica actualmente en las herramientas existentes relacionadas con la dosificación de precisión, si es que utiliza alguna o está en conocimiento de su funcionamiento? **(Respuesta libre)**

6.2.2.3. Resultados

1. ¿Le pareció innovadora la idea de aplicar gemelos digitales en la salud? **Muy innovadora / Moderadamente innovadora / Poco innovadora / Nada innovadora**

2. ¿Qué tan importante considera que fue la integración de datos en tiempo real desde dispositivos como wearables? **(del 1 al 5, donde 1 es "Nada importante" y 5 es "Muy importante")**

3. ¿Qué tan relevante es para usted que una plataforma sea fácil de usar en la práctica médica diaria? **(del 1 al 5, donde 1 es "Nada relevante" y 5 es "Muy relevante")**

4. ¿Considera que la plataforma presentada es intuitiva y sencilla de utilizar? **Sí / Normal / No**

5. ¿Considera que la plataforma presentada es visualmente agradable? **Sí / Normal / No**

6. ¿Cuáles funcionalidades considera prioritarias en esta plataforma? **Extensión de cantidad de posibles modelos a utilizar para la dosificación de precisión / Monitoreo en tiempo real / Gestión de pacientes / Interfaz web amigable**

6.2.2.4. Validaciones

1. ¿Utilizaría esta plataforma si fuera un médico o químico involucrado? **Sí / No / Otro:** *(justifique opcionalmente)*

2. ¿Considera que es una plataforma segura? **Sí / No lo sé / No**

6.2.2.5. Mejoras

1. ¿Piensa que se podría haber agregado alguna otra funcionalidad más importante que las integradas? Si es así, ¿cuál? *(Respuesta libre)*

2. ¿Tiene alguna sugerencia en cuanto a la usabilidad de la plataforma? *(Respuesta libre)*

3. ¿Tiene alguna otra opinión acerca de mejoras o correcciones de la plataforma? *(Respuesta libre)*

6.2.3. Resultados de la encuesta

La encuesta tiene una participación diversa, con profesionales como docentes, ingenieros, químicos farmacéuticos y estudiantes de doctorado, lo que enriquece las perspectivas recopiladas. Una mayoría significativa de los encuestados tiene relación directa con la medicina de precisión o tecnología de la salud, lo que refuerza la relevancia de sus opiniones para evaluar la plataforma. En total, se recaban siete respuestas.

El concepto de gemelos digitales es bastante conocido por los participantes, quienes lo calificaron como «muy innovador», excepto una respuesta que lo consideró «moderadamente innovador». Esto sugiere que la idea tiene potencial para generar interés, aunque también hay espacio para trabajar en su percepción.

En cuanto a la importancia de la personalización en tratamientos médicos, la puntuación promedio fue alta, con casi todas las respuestas dándole el puntaje

más alto. De igual manera, hay consenso en que los gemelos digitales podrían mejorar la dosificación de precisión y que integrar datos en tiempo real desde dispositivos como *wearables* es bueno para maximizar su utilidad.

La usabilidad de la plataforma fue valorada de forma positiva por la mayoría. Se destacó su diseño intuitivo y amigable, aunque algunos mencionaron oportunidades para simplificar aún más la interacción, como incluir menús desplegables para evitar errores en la introducción de datos. También hubo menciones a la necesidad de reducir distracciones visuales, como el fondo de constelaciones en la interfaz en la pantalla del paciente.

En términos de funcionalidades, los participantes mencionaron la importancia de incluir la posibilidad de tener modelos para dosificación, monitoreo en tiempo real, gestión de pacientes, y herramientas que permitan la comparación de simulaciones. Algunos sugirieron mejoras, como integrar la plataforma con la historia clínica y garantizar la interoperabilidad con otros sistemas comunes.

Finalmente, la seguridad fue una preocupación recurrente. Aunque los encuestados no identificaron problemas específicos en esta área, se propusieron medidas como autenticación avanzada y estrategias para la impersonalización de datos sensibles.

En base a todas las respuestas, en general la plataforma fue considerada útil, innovadora y bien diseñada, y hay algunas recomendaciones para fortalecer su funcionalidad y usabilidad. En el Anexo Validaciones se encuentran todas las respuestas a la encuesta.

Capítulo 7

Etapas y gestión de proyecto

En este capítulo se detallan las principales etapas del proyecto y se describe cómo se realiza su gestión.

7.1. Cronograma inicial y estimaciones

Al comienzo del proyecto se planifica un cronograma a modo de borrador, estimando posibles duraciones y etapas que el proyecto iba a presentar. Al iniciar el cronograma en setiembre del año 2023, se toma en cuenta un mes de descanso (el mes de Enero del año 2024).

El cronograma inicial se piensa mes a mes para las etapas que ya se podían planificar y luego se estiman las grandes etapas restantes. El cronograma inicial planeaba seguir las etapas detalladas a continuación:

7.1.1. Primer mes (Setiembre 2023)

Se estima que el primer mes será de estudio del marco conceptual y conocimiento existente en el área. Se investigará sobre dosificación de precisión informada por modelos, Gemelos Digitales, Gemelos Digitales aplicados al área médica, el uso de *wearables* para la dosificación de precisión y la historia clínica electrónica de Uruguay.

7.1.2. Segundo mes (Octubre 2023)

El segundo mes será de seguir investigando y comenzar a analizar la problemática planteada. Se leerá documentación de la plataforma existente (Finglix) y se investigarán plataformas similares. Además se comenzará a investigar posibles aplicaciones de Gemelos Digitales a la plataforma existente y se tendrán charlas con los clientes para relevar requisitos, entender los puntos más débiles

de la plataforma y comparar las ideas de los clientes con lo investigado anteriormente para tomar la decisión de la viabilidad de sus pedidos.

7.1.3. Tercer mes (Noviembre 2023)

El tercer mes se dedicará al análisis de alternativas arquitectónicas y tecnológicas para la solución a desarrollar. Se llevará a cabo una búsqueda, un análisis y una comparación entre plataformas de Gemelos Digitales para luego inclinarse por alguna y se diseñará a grandes rasgos la arquitectura general y de cada parte del proyecto, así como las tecnologías a usar.

7.1.4. Cuarto mes (Diciembre 2023)

El cuarto mes será de definir totalmente el alcance del proyecto y de finalizar con el diseño de la arquitectura y la toma de decisiones de diseño.

7.1.5. Quinto y sexto mes (Febrero 2023 - Marzo 2023)

El quinto y sexto mes se desarrollará la propuesta de solución. Se hará iterando según el *feedback* de las tutoras del proyecto y según el *feedback* de los clientes.

7.1.6. Séptimo y octavo mes (Abril 2023 - Mayo 2023)

Los meses séptimo y octavo se dedicarán a realizar la validación y verificación del producto. Se harán los cambios finales según la devolución del cliente y se escribirán casos de prueba. Además se comenzará a escribir el informe final.

7.1.7. Noveno mes (Junio 2023)

El noveno mes se dedicará totalmente a la documentación de la solución. Se detallará el marco teórico, la descripción de la solución encontrada (diseño e implementación), se mencionarán las decisiones tomadas, conclusiones y posibles mejoras a futuro.

7.2. Comunicación

A lo largo del proyecto las integrantes se reúnen de forma virtual y presencial. Se tienen reuniones presenciales al menos una vez por mes y luego dos reuniones semanales virtuales. Además, se tiene una comunicación diaria constante de las integrantes mediante WhatsApp para así despejar dudas rápidas y hacer más ágiles las reuniones virtuales. Se utiliza GitHub como otro medio de comunicación para las buenas prácticas del código, permitiendo revisar el código de la otra integrante y dejar comentarios de retroalimentación y fomentando la discusión para un código limpio.

Para la comunicación con las tutoras se utiliza el correo electrónico y Zoom para las reuniones sincrónicas. Se intenta realizar reuniones quincenales con las tutoras pero de ser necesario se adelanta o se pospone alguna. En las primeras etapas del proyecto, que eran más enfocadas a lo conceptual y teórico, se realizan presentaciones de diapositivas e informes indicando los avances de esa quincena a las tutoras. Se muestran los documentos e informes realizados en ese tiempo y el trabajo con el que se iba a continuar.

La comunicación con el cliente final se tiene dos veces directamente y luego mediante las tutoras como intermediarias. Se realiza una reunión cercana al comienzo del proyecto con el cliente para poder relevar sus requerimientos, viendo sus necesidades y opiniones. Posteriormente, se envían videos a modo de demostración por correo electrónico a las tutoras para que ellas los reenvíen a los clientes, con el fin de que validen los flujos y pantallas.

7.3. Herramientas utilizadas

En esta sección se listan y se explica la funcionalidad de las herramientas utilizadas a lo largo del proyecto.

7.3.1. Github

GitHub es una plataforma de desarrollo colaborativo que utiliza Git, un sistema de control de versiones, para facilitar la gestión de proyectos de software. En GitHub, los desarrolladores pueden almacenar su código en repositorios y colaborar con otros, permitiendo un seguimiento de los cambios realizados en el código a lo largo del tiempo, lo que mejora la colaboración en equipo.

Además, GitHub ofrece características como la gestión de problemas (issues) para rastrear errores y tareas, y la posibilidad de realizar *pull requests* para que otros revisen y fusionen el código. También incluye herramientas para la documentación y la automatización de tareas a través de GitHub Actions. (Github, s.f.)

7.3.2. Visual Studio Code

Visual Studio Code es un editor de código fuente desarrollado por Microsoft, diseñado para ser ligero y versátil. Ofrece una amplia gama de funciones que facilitan el desarrollo de software, incluyendo resaltado de sintaxis, autocompletado de código y depuración integrada. Visual Studio Code también cuenta con una interfaz amigable que admite múltiples paneles, lo que facilita el trabajo en proyectos complejos. Además, se integra con sistemas de control de versiones como Git, lo que permite gestionar el código y colaborar con otros de manera eficiente. (VSCode, s.f.)

7.3.3. Google Drive

Google Drive es un servicio de almacenamiento en la nube desarrollado por Google que permite a los usuarios guardar, compartir y colaborar en archivos de manera sencilla. Ofrece una gran variedad de aplicaciones, incluyendo Google Docs, Google Slides y Google Sheets, que permiten crear y editar documentos, presentaciones y hojas de cálculo, respectivamente. Con Google Drive, los usuarios pueden trabajar en sus archivos en tiempo real, lo que facilita la colaboración entre equipos, ya que múltiples personas pueden editar el mismo documento simultáneamente. Además, todos los cambios se guardan automáticamente, lo que reduce el riesgo de perder información. La plataforma también permite compartir archivos con facilidad, estableciendo diferentes niveles de acceso para los colaboradores, desde solo visualización hasta edición completa. (Google Drive, s.f.)

7.3.4. WhatsApp

WhatsApp es una aplicación de mensajería instantánea que permite a los usuarios enviar mensajes de texto, realizar llamadas de voz y video, y compartir imágenes, videos y documentos de manera rápida y sencilla. (Whatsapp, s.f.)

7.3.5. Mail

El correo electrónico es un servicio de comunicación digital que permite a los usuarios enviar y recibir mensajes a través de Internet. Este sistema facilita la transmisión de información en forma de texto, así como la posibilidad de adjuntar archivos como documentos, imágenes y videos.

7.3.6. Zoom

Zoom es una plataforma en línea destinada a las comunicaciones a distancia, permitiendo a los usuarios realizar reuniones virtuales. Ofrece varias funciones como compartir pantalla, chats en tiempo real, grabación de las reuniones, entre otras. (Zoom, s.f.)

7.3.7. Mattermost

Mattermost es una plataforma de comunicación y colaboración en equipo que permite enviar mensajes, compartir archivos y organizar proyectos. Se pueden crear canales para diferentes temas, enviar mensajes directos y hacer videollamadas. (Mattermost, s.f.)

7.3.8. Overleaf

Overleaf es una plataforma en línea que permite a los usuarios crear, editar y colaborar en documentos usando LaTeX, un sistema comúnmente utilizado para textos científicos, matemáticos y técnicos. Lo práctico de Overleaf es que hace

que LaTeX sea más accesible mediante una interfaz intuitiva en el navegador, sin necesidad de instalar software adicional.

Una de sus características más destacadas es la posibilidad de colaborar en tiempo real, lo que permite a varios usuarios trabajar en el mismo documento simultáneamente. Además, ofrece una vista previa automática mientras se edita el código LaTeX, lo que facilita ver los resultados al momento. (Overleaf, s.f.)

7.3.9. Lucidchart

Lucidchart es una herramienta en línea que permite hacer diagramas de manera sencilla y rápida. Se pueden crear organigramas, diagramas de flujo, mapas mentales, entre otros. Además provee colaboración con otras personas en tiempo real, lo que facilita el trabajo en equipo. (Lucid Chart, s.f.)

7.3.10. Visily - WireFrames

Visily es una herramienta en línea que permite crear prototipos y diagramas visuales como *wireframes* de forma sencilla. Está enfocada para diseñadores y desarrolladores que quieran visualizar ideas y conceptos antes de llevarlos a cabo. Se pueden arrastrar y soltar componentes para construir interfaces y diagramas, y se puede colaborar en tiempo real con un equipo. Visily también ofrece plantillas y elementos predefinidos para poder basarse en ellos. (Visily, s.f.)

7.3.11. Elastic Cloud - Antel

Elastic Cloud, provisto por el servicio Mi Nube de Antel, es una solución de nube que permite a las empresas y desarrolladores implementar y gestionar soluciones de búsqueda, análisis y visualización de datos utilizando la tecnología de Elasticsearch, Kibana y otros productos de la línea Elastic. Este servicio ofrece la capacidad de escalar y personalizar aplicaciones basadas en datos en tiempo real, facilitando la búsqueda de información y el análisis de grandes volúmenes de datos.

Elastic Cloud proporciona una infraestructura segura y confiable, con características como la gestión automática de actualizaciones, copias de seguridad y monitoreo de rendimiento. Además, permite a los usuarios acceder a sus datos desde cualquier lugar y en cualquier momento, lo que mejora la colaboración y la toma de decisiones basada en datos. (Elastic Cloud Antel, s.f.)

7.4. Metodología de trabajo

En esta subsección se justifica y explica la elección tomada de adoptar un enfoque ágil iterativo incremental para el desarrollo del proyecto.

Atlassian es una de las grandes empresas de software mundiales dedicadas al desarrollo de herramientas de colaboración y gestión para equipos. Posee gran

parte de las herramientas utilizadas con este fin, desde Jira (seguimiento de tareas), Bitbucket (alojamiento de repositorios Git), e incluso Trello (tableros y tarjetas). Esta empresa, en su web oficial (Metodología ágil, s.f.), define la metodología ágil como un concepto de gestión de proyectos que implica dividir al mismo en fases, un equipo colaborativo y la aplicación de mejoras continuas. En la misma línea, Abuchar Porras (2023) menciona que los equipos deben seguir un ciclo de planificación, ejecución y evaluación. Como beneficio, se hace hincapié en la flexibilidad de esta metodología para adaptarse y responder ante los cambios en el mercado o ante el *feedback* de los clientes de forma rápida. La planificación justa y el desarrollo de pequeños incrementos de forma frecuente permiten al equipo tener devoluciones sobre cada cambio e integrarlos en los planes a futuro.

El libro, menciona que el método ágil, tiene los siguientes principios:

- **Individuos e interacciones** sobre procesos y herramientas.
- **Software funcionando** sobre documentación extensiva
- **Colaboración con el cliente** sobre negociación contractual
- **Respuesta ante el cambio** sobre seguir un plan

Debido a que el proyecto se basa en poseer una plataforma amigable y que aporte resultados y pueda ser utilizable por médicos, el constante *feedback* es importante y adaptarse a él aún más. Se centra en la colaboración con el cliente y sus necesidades. Esto conlleva a centrarse en un *software* que funcione más que documentación que abunde. Por lo tanto, se toma la decisión de que en este caso, lo mejor es utilizar un enfoque ágil iterativo incremental.

7.5. Etapas reales del proyecto

El proyecto se desarrolla en cinco grandes etapas y tiene una duración aproximada de 13 meses. Las primeras etapas ocurren de forma consecutiva, dado que se enfocan en aspectos teóricos y de investigación. Posteriormente, las etapas comienzan a superponerse y a realizarse en paralelo, lo cual hace que la duración total no corresponda a la suma directa del tiempo de cada una de ellas.

7.5.1. Inicio del proyecto y primer análisis

El proyecto comienza en Setiembre del año 2023, con el objetivo de mejorar una plataforma existente que se basa en la dosificación de precisión informada por modelos. Comienza con la idea de implementar la tecnología de Gemelos Digitales en la misma y además incorporarle una interfaz visual más amigable para que sea sencilla de utilizar por distintos tipos de usuarios como químicos, médicos y pacientes.

Se inicia relevando información acerca de Gemelos Digitales. Se investiga su concepto, para qué se utilizan y su posible vínculo con el área médica. Se buscan proyectos médicos que implementen esta tecnología y la forma en la que la implementan. Dentro de las formas vistas de utilización de esta tecnología, se discuten opciones para implementar en la solución existente.

Se realizan varias reuniones con las tutoras para comprender el modelo conceptual de las posibles soluciones y decidir el enfoque del proyecto.

Esta etapa finaliza luego de una reunión junto al equipo que trabaja sobre la solución Finglix 2.0 y con el cliente. En la reunión se obtienen las necesidades de los clientes y el enfoque que le desean dar al proyecto. Validan la idea presentada de crear un Gemelo Digital por paciente y que este se conecte con *wearables* del paciente para ir obteniendo información en tiempo real. El equipo de clientes menciona los modelos PB/PK y Mrgsolve como herramientas interesantes para agregar al proyecto y no sólo poder simular con los modelos que existen en la plataforma previa.

Esta etapa es importante para el equipo ya que el conocimiento previo del dominio no abundaba. Al finalizar esta etapa se tiene un mejor manejo de los conceptos a utilizar en el marco del proyecto.

Duración: 4-5 meses

7.5.2. Análisis, relevamiento y objetivos del proyecto

En esta etapa se inicia con el relevamiento de requerimientos tanto funcionales como no funcionales de la plataforma. Se tienen múltiples reuniones con las tutoras para definir los mismos. Se realizan flujos con *wireframes* para proponer la paleta de colores y los diseños principales de la parte web del proyecto y así poder validar estos aspectos con el cliente.

Además, se realizan los diagramas conceptuales de la solución, el diagrama UML y los diagramas de componentes y despliegue de la solución. Estos diagramas se modifican con el correr de las reuniones con las tutoras para así ajustarlos definitivamente a los requerimientos relevados.

Se decide realizar una aplicación móvil en lugar de conectar un *wearable* específico por paciente ya que permite una mayor variedad de datos y es más útil hoy en día ya que casi todo paciente posee un celular inteligente. En este período también se decide utilizar Eclipse Ditto como plataforma para los datos en tiempo real del paciente.

Se define el alcance del proyecto teniendo en cuenta los requerimientos relevados y la dificultad de cada tarea y flujo a realizar. También se define el flujo de trabajo a seguir en cuánto al tablero de tareas, la paralelización de las mismas, la revisión de código y la documentación a generar en cada etapa. Se decide ir documentando en el proceso para que no haya información olvidada y que el proceso de documentación se agilice sobre el final.

Finalmente se configura el backend y frontend, generando su respectiva documentación y detalles de las decisiones tomadas en cuánto a la arquitectura. Se toman decisiones siguiendo las buenas prácticas y el tamaño y enfoque del proyecto.

Duración: 2 meses

7.5.3. Construcción del producto

En esta etapa se construye el producto. Se comienza con el desarrollo en paralelo del backend y frontend con las primeras funcionalidades previas a la conexión con un proveedor de simulación (registro de pacientes, médicos, inicio de sesión, listado de pacientes, cargar datos básicos del paciente, entre otros).

Se investiga cómo vincular Mrgsolve con el proyecto y se realiza la conexión mediante una biblioteca que permite ejecutar código R en javascript. Se buscan modelos ya creados en C++ para poder utilizar con Mrgsolve y se agregan al proyecto. Paralelamente, se continúa investigando acerca de Eclipse Ditto y creando la aplicación móvil a modo de conexión con los datos en tiempo real del paciente.

Se hace el despliegue de Eclipse Ditto en la nube de Antel para facilitar su uso y que no se tenga la necesidad de ejecutarlo localmente cada vez.

Por el lado de la aplicación, se realizan los flujos de las simulaciones, tratamientos e ingreso de datos, para los modelos cargados de Mrgsolve. Se vincula completamente la solución, integrando la aplicación móvil con Ditto y con la aplicación web mediante la API brindada por Eclipse Ditto. En las reuniones quincenales con las tutoras se realizan demostraciones en vivo de los avances y con los consejos de ellas se va mejorando y ajustando la solución.

Finalmente se realiza una reunión con el equipo que trabaja en la plataforma anterior (Finglix 2.0) para poder realizar la conexión con la prlataforma desarrollada por ellos y agregar Finglix como otro proveedor de simulación además de Mrgsolve.

Duración: 5 meses

7.5.4. Despliegue y validación del producto

En esta etapa se hace el despliegue de la plataforma web y del servidor *backend*. Primero se investiga cómo desplegar aplicaciones en la nube de Antel y cuáles son las opciones para hacerlo. Se investigan varias opciones para el *frontend* pero se decide por *nginx*. De esta forma, se realiza el despliegue de la aplicación web en la nube de Antel y se corrobora que quede todo instalado. Luego se investiga la forma de despliegue de servidores existentes en la nube y se decide realizarlo mediante *Docker*. Por lo tanto, se realiza el despliegue del *backend*. Este despliegue es el que se considera más difícil, ya que implica desplegar código en *R*, la base de datos del servidor y el panel de administración. Luego de varias pruebas y errores que se generan por incompatibilidades de versiones de paquetes, se logra desplegar correctamente. Por otro lado, la aplicación móvil se puede instalar mediante su APK (generado en esta etapa) en celulares Android.

De forma paralela a los despliegues se realiza una validación del producto con investigadores y estudiantes de posgrado del grupo GIDP e integrantes del LINS para poder evaluar la solución. De esta forma se obtienen opiniones, sugerencias y aportes a futuro para la plataforma. Esta validación se dicta mediante una

presentación, por parte de las integrantes de este proyecto, del contexto del mismo y una explicación en forma general de sus funcionalidades y objetivos. Esto fue complementado con una *demo* de la plataforma para que se visualicen de mejor forma los flujos. Se concluyó una hora y media de reunión incluyendo la presentación, *demo* y preguntas y sugerencias finales. Al finalizar se le envió un formulario a los presentes en la reunión para obtener *feedback* y sugerencias acerca de la solución.

Duración: 2 meses

7.5.5. Cierre y Documentación

En esta etapa se toma como base el *template* de documento de proyecto final de grado y se comienzan a agregar los documentos que se fueron creando a lo largo del proyecto. Por esto es que el tiempo marcado para esta etapa (2 meses) es relativo, ya que una gran base de la documentación ya se había escrito al lo largo de las otras etapas.

Primero se decide la estructura general del proyecto y luego en cada capítulo se van agregando los documentos correspondientes.

Luego, se va releyendo y modificando lo necesario, así como completando el informe con los capítulos restantes. A medida que se escribe un capítulo se van haciendo lecturas para iterar y así encontrar errores y mejoras a realizar.

Se escriben además los manuales de usuario y anexos necesarios para la utilización de la plataforma.

El siguiente paso es la presentación y defensa del proyecto en la Facultad de Ingeniería de la Universidad de la República.

Duración: 2 meses

7.6. Comparación cronograma inicial y real

En las secciones 7.1 y 7.5 se presentan los cronogramas iniciales y los reales del proyecto. En esta sección se comparan los cronogramas y comentan las principales diferencias.

La primer diferencia entre los cronogramas está en el tiempo destinado al marco teórico y comprender conceptualmente todos los elementos y términos involucrados en el proyecto. Para poder brindar una solución lo más integral posible, se decidió tomar el tiempo necesario para la investigación y aprendizaje teórico sobre todos los temas involucrados.

Como segundo punto a observar es que el análisis de requerimientos del cliente tomó mayor tiempo. Se requirieron varios intercambios con el cliente hasta aclarar realmente sus necesidades y sus ideas en cuanto al proyecto. Se logra llegar a un alcance en común que sería viable realizar, pero para llegar a esta conclusión en común se itera por varias posibles soluciones.

Otro punto a observar en la comparación de los cronogramas, es que se ve que se han hecho más funcionalidades y plataformas de las que en un principio se esperaban. Como ejemplos claros, se han vinculado nuevas bibliotecas para

simular que soportan distintos tipos de modelos (Mrgsolve) y se ha creado una aplicación móvil que se ha vinculado en tiempo real con la plataforma. Esto ha incrementado el tiempo de desarrollo de la solución en gran medida pero también ha permitido crear una solución más integral.

Finalmente, las etapas de validación y verificación, así como la de cierre y documentación, no resultan tan desviadas en su estimación, ya que llevan aproximadamente dos meses cada una, aunque se realizan en paralelo. Si se realizaran en secuencia, probablemente se ajustarían a la estimación general.

Capítulo 8

Conclusiones y Trabajo Futuro

En este capítulo se examinan los resultados obtenidos junto con las dificultades enfrentadas a lo largo del proyecto. Se comparan los objetivos planteados inicialmente con los logros alcanzados, se analizan los aportes realizados y se sugieren líneas futuras de desarrollo, prioridades y áreas clave para continuar el avance de este trabajo.

8.1. Conclusiones generales

El desarrollo de esta plataforma basada en Gemelos Digitales en el ámbito de la dosificación de precisión cumple con los objetivos planteados y proporciona una solución innovadora para la medicina personalizada.

El análisis sobre las aplicaciones de los Gemelos Digitales en el área médica sienta las bases para identificar casos de uso para dosificación de precisión, además de explorar la integración con *wearables*. Este conocimiento permite orientar el desarrollo de la plataforma hacia aplicaciones prácticas y con un impacto potencial en la personalización de tratamientos.

Por otra parte, el análisis de la problemática que se realiza es fundamental para entender a fondo las necesidades de los usuarios finales y diseñar una solución que responda a ellas. Este proceso incluye el relevamiento de requerimientos tanto funcionales como no funcionales, buscando un diseño alineado no solo con el entorno de dosificación de precisión, sino también con la flexibilidad necesaria para ser aplicado en distintos ámbitos y centros de salud.

El diseño de una plataforma extensible, uno de los objetivos clave del proyecto, resulta en una arquitectura modular y flexible. Gracias a esto, se confirma que la plataforma puede adaptarse a diferentes necesidades y contextos, potenciando su valor en el área de la salud.

La implementación de un prototipo funcional valida la viabilidad del diseño propuesto, cumpliendo con los requerimientos establecidos. Este prototipo ini-

cial permite tener una versión que sirve como base para mejoras y extensiones futuras.

Finalmente, las pruebas unitarias, de integración y de código permiten evaluar la calidad de la plataforma. El proceso de evaluación es reutilizable en caso de que se quieran agregar nuevos módulos o funcionalidades, y es esencial para garantizar la estabilidad del sistema en cuanto a funcionalidad y calidad.

El proyecto genera una plataforma basada en Gemelos Digitales que no solo cumple con los requisitos específicos de dosificación de precisión, sino que también está diseñada para adaptarse a un ecosistema de salud digital en rápida evolución. Con su arquitectura con foco en la extensibilidad y su enfoque en la experiencia de usuario, esta herramienta puede ser una solución interesante para generar un mayor nivel de personalización en la medicina.

8.2. Mayores desafíos

En esta sección se mencionan los mayores desafíos encontrados a lo largo del proyecto.

8.2.1. Uso y despliegue de Eclipse Ditto

La configuración de Eclipse Ditto fue un gran desafío, ya que para poder utilizarlo se requiere un estudio previo extenso acerca de conceptos relacionados a la plataforma. Además, la documentación no es demasiado clara y parece estar orientada a profesionales con experiencia en IoT. Este no es el caso de las integrantes del equipo, por lo que resultó muy difícil entender cómo realizar las configuraciones necesarias, gestionar *things*, *policies*, crear conexiones MQTT y gestionar el *broker*, entre otras cosas.

El despliegue mediante Kubernetes también requiere una gran cantidad de investigación y conocimiento previo. Considerando la documentación escasa y el alto consumo de recursos de la plataforma, tuvo un alto costo para el proyecto poder lograr ejecutar el servidor en la nube de forma de que funcione de la manera esperada. En particular, no fue fácil encontrar información de que la razón por la cual no estaba funcionando era que no tenía suficientes recursos asignados y de cómo solucionarlo.

8.2.2. Integración del lenguaje R con Node.js

El paquete Mrgsolve fue una de las herramientas solicitadas por los clientes para integrar en el proyecto, ya que permite realizar simulaciones avanzadas con modelos PK/PD, PB/PK y otras variantes que Finglix no cubría. Al decidir incorporarlo, se requirió investigar cómo hacerlo, dado que Mrgsolve es un paquete en R, mientras que el *backend* está desarrollado en Node.js con JavaScript.

El primer desafío fue entender cómo ejecutar lenguaje en R dentro de JavaScript. Para esto, se encontró la biblioteca *r-integration* que con ciertos comandos

permite ejecutar *scripts* en R desde NodeJS por lo que el primer obstáculo estaba resuelto.

Luego, se tuvo que estudiar la documentación de Mrgsolve y de R para comprender cómo escribir un *script* en R, y cómo simular modelos dentro del mismo y obtener resultados. La documentación de Mrgsolve y de R es detallada, por lo que este proceso llevó el tiempo de lectura de la documentación. Mientras se comprendía el tema se debió instalar R Studio para poder probar cómo se corre el código en R e ir investigando qué se podía hacer con modelos y Mrgsolve. R Studio es una aplicación que permite ejecutar código en R y guarda y muestra resultados y datos extraídos. Con estas pruebas y la lectura de la documentación se fue escribiendo el código tanto para simular un modelo y para obtener las covariables de los mismos (son los dos casos en los que se quiere ejecutar código para extraer datos del modelo).

Dentro de la creación de los *scripts*, lo más difícil fue pasar parámetros desde JavaScript hacia el *script* en R y pasar resultados desde R hacia JavaScript. Mediante prueba y error, se pudo resolver que se deben pasar todos los parámetros en formato *string* por lo que se debió hacer un JSON.stringify a los objetos dado que los lleva a un formato de caracteres. Luego, desde el script de R se deben extraer y volver a pasar a un formato comprensible por el lenguaje, realizando la correspondencia según sus datos y valores.

A su vez, luego de haber podido ejecutar los modelos con los datos que se le pasaban, se tuvo que ver de qué forma devolvía los datos Mrgsolve y de qué forma extraer solo los datos que se necesitaban. Para esto se investigaron a fondo varias funciones de R, como la de obtener el valor máximo de un vector, construir listas, reemplazar *strings* por otras y combinar dos listas, entre otras.

Finalmente, cabe mencionar que otra gran dificultad fue el *debug* cuando se obtenían errores en la ejecución de algún modelo. Esto se debe a que al capturar el error en el *script* de R y devolverlo, éste se envía en formato *string* por lo que se pierden la mayoría de datos y crea un texto grande solamente con símbolos y no tiene ningún texto claro del error que se produjo.

8.2.3. Diseño de solución que considere uso de wearables

Otro desafío del proyecto consistió en considerar el uso de *wearables* en la solución. La manufactura de *wearables* no está tan estandarizada como, p. ej., el desarrollo móvil. Existen grandes cantidades de sistemas operativos de código propietario para los distintos tipos de *wearables*, por lo que abarcar la mayor cantidad posible de dispositivos requería un análisis profundo. Se arribó a la solución del uso de dispositivos móviles después de evaluar las limitaciones y la fragmentación existente en el ecosistema de *wearables*.

La decisión se fundamentó en que los dispositivos móviles ofrecen una plataforma más unificada y accesible, con sistemas operativos ampliamente compatibles y mejores herramientas de desarrollo. Además, estos dispositivos permiten integrar los datos de salud de manera continua y confiable, lo cual es clave para la dosificación de precisión.

8.3. Análisis retrospectivo

En esta sección se realiza un análisis retrospectivo de algunas decisiones adoptadas a lo largo del desarrollo del proyecto, reflexionando sobre los resultados obtenidos y el impacto de dichas decisiones en el producto final.

8.3.1. Eclipse Ditto para implementar Gemelos Digitales

A pesar de que inicialmente Ditto parecía una solución ideal para agregar Gemelos Digitales al proyecto, la integración resultó bastante más costosa de lo esperado, el consumo de recursos no es muy eficiente y los beneficios no son tan claros.

Si bien, de acuerdo a lo que se promocionaba, no hubo que escribir código para integrar Eclipse Ditto al proyecto, la configuración y despliegue fueron muy costosos, como se menciona en la sección 8.2.1. Además, las ventajas proporcionadas por la plataforma no son claras en el caso de uso de este proyecto. Al contar con un *backend*, las *things* podrían modelarse como atributos dentro de la entidad Paciente. En las etapas iniciales del proyecto se analizó la posibilidad de no contar con un backend, pero se descartó debido a la necesidad de implementar usuarios para los médicos, centralizar los proveedores de simulación, entre otras cosas. Si el proyecto no contara con un *backend*, Ditto es una solución interesante ya que permite crear entidades con soporte para actualizaciones en tiempo real.

Otro factor que influyó en la inclinación hacia Ditto es que existió una mala interpretación de las capacidades de la plataforma. En un momento se pensó que Ditto permitía guardar todo el historial de datos registrados en su base de datos no relacional. Esto resultaba prometedor a la hora de realizar simulaciones como las de tipo PB/PK, que pueden necesitar consumir altas cantidades de datos y guardarlas en la plataforma resulta mucho más escalable que guardarlas en una base de datos relacional. En este escenario, Ditto presentaba un atractivo muy grande. Sin embargo, en una etapa de madurez alta del proyecto se descubrió que no se guarda un histórico de los datos por defecto. Es posible crear una integración con Kafka para lograr este comportamiento, pero el alcance y la complejidad ya existente del proyecto llevaron a que se descarte esta posibilidad y se deje como trabajo futuro.

Finalmente, Ditto consume muchos recursos. El código promocional de Elastic Cloud no es suficiente para mantener la infraestructura corriendo todos los días durante un mes, por lo que se decide «apagar» el servidor mientras no se está usando.

Por todas estas razones, se llega a la conclusión de que implementar una solución propia utilizando *sockets* o incluso el *broker* MQTT de Eclipse conectado al *backend* hubiera resultado más eficiente con respecto al tiempo y consumo de recursos.

8.3.2. Integración de Mrgsolve

Si bien la integración de esta herramienta como proveedor de simulaciones es muy beneficiosa en el contexto del proyecto, resulta un poco redundante considerando la integración con Mrgsolve realizada como parte del trabajo de Finglix 2.0. La razón de la implementación de esta herramienta en ambas plataformas se debe a que los proyectos se realizaron simultáneamente. Gracias a esta implementación, los mismos modelos pueden incorporarse tanto en este proyecto como en Finglix y utilizarse en MedFing. Sin embargo, uno de los beneficios adicionales de integrar Mrgsolve en el *backend* de la solución descrita en este documento es que permite ilustrar la naturaleza flexible de la plataforma MedFing, así como su capacidad para operar de forma agnóstica respecto al proveedor de simulación.

8.4. Trabajo a futuro

En esta sección se listan los posibles trabajos a futuro agrupados por categoría.

8.4.1. Integración con sistemas externos

En relación a la integración con otros sistemas se plantea la integración con el usuario **gub.uy** para la autenticación de la aplicación móvil. Además, también resulta interesante la integración con la **historia clínica electrónica** de cada paciente como fuente de datos. De esta forma el médico podría tener el historial completo y actualizado del paciente. Otra integración que aporta al manejo de datos de la solución es con **Finglix como proveedor de datos** para poder obtener la información de simulaciones directamente realizadas en la plataforma Finglix. Por último, la integración con los términos médicos estándares de **SNOMED** para la toma de datos es útil ya que de esta forma los diagnósticos son estándares y fáciles de comprender por los médicos. Esto facilita la carga y lectura de datos del paciente.

8.4.2. Integración de nuevos componentes

En relación a la integración de nuevos componentes se presenta la integración de la aplicación móvil con celulares **iOS**. Esto aumenta la cantidad de pacientes con posibilidad de tener su Gemelo Digital conectado con el médico. Otra integración planteada es de Eclipse Ditto con **Kafka** o un servicio similar que permita guardar el historial de todos los datos del paciente. Por úlitmo, se propone integrar el paquete de **R** `Mapbay` ya que esto personalizaría aún más las simulaciones, optimizando los modelos que se usan en Mrgsolve.

8.4.3. Mejoras generales de la plataforma

En esta sección se plantean mejoras generales de la plataforma.

Modelos

En cuanto a los modelos, se propone la carga de un modelo directamente desde la página de administración sin necesidad de agregarlo directamente en el repositorio. Además, la especificación en la carga de los datos que se desean graficar del modelo y el tiempo de simulación (tiempo de llegada al estado estacionario de cada droga). Otra mejora sería la carga de las covariables del modelo al agregarlo, así estos se guardan en la base de datos y no se tiene que correr el modelo entero cada vez que se requiere obtener sus covariables. Finalmente, se menciona la carga del rango normal de concentración de la droga esperado y saludable. De esta forma se podría indicar desde la plataforma si está por debajo o sobre el máximo de lo deseado y de lo indicado como no riesgoso para esa droga.

Tratamientos

En cuanto a los tratamientos, se propone que dentro de un mismo tratamiento, antes de comenzado, se pueda simular con varios proveedores y no solamente con uno. Además que dentro de un mismo tratamiento, antes de comenzado, se pueda realizar la misma simulación pero con distinta droga para comparar y elegir cuál droga es la indicada. Por úlitmo se plantea que luego de que un tratamiento comenzó, se pueda ir agregando los datos de las concentraciones reales que va obteniendo el paciente y así poder ir comparando con lo graficado en las simulaciones.

Pacientes

En cuanto a los pacientes, se propone la creación de pacientes con los datos de los pacientes de una mutualista dada. Además se plantea la sincronización de datos de los pacientes de una mutualista con usuarios ya existentes en la plataforma para mantener los datos actualizados. Como último aspecto de pacientes se menciona la estandarización de los íconos entre la aplicación web y la aplicación móvil.

Validaciones

Se propone para las validaciones, agregar la validación de la solución propuesta con usuarios reales del ámbito de la salud.

Misceláneo

En cuanto a aspectos extras, se menciona la traducción de las plataformas al español.

Referencias

Abuchar Porras, A. (2023). *Metodologías ágiles para el desarrollo de software*. Editorial Universidad Distrital Francisco José de Caldas. Descargado de https://books.google.com.uy/books?id=JfXBEAAAQBAJ

AdminJS. (2024). *Adminjs - the leading open-source admin panel for node.js apps — adminjs*. Descargado de https://adminjs.co/ (Último acceso: 24-10-2024)

AGESIC. (s.f.). *Salud digital*. https://www.gub.uy/agencia-gobierno-electronico-sociedad-informacion-conocimiento/politicas-y-gestion/programas/es-saluduy. (Último acceso: 26-09-2024)

AGESIC. (2022). *¿qué es id uruguay?* Descargado de https://www.gub.uy/agencia-gobierno-electronico-sociedad-informacion-conocimiento/politicas-y-gestion/es-id-uruguay (CÚltimo acceso: 4-12-2024)

Ahmadi-Assalemi, G., Al-Khateeb, H., Epiphaniou, G., Alhaboby, Z. A., Maple, C., Alkaabi, S., y Alhaboby, D. (2020). *Cyber defence in the age of ai, smart societies and augmented humanity* (H. J. et al., Ed.). Switzerland: Springer Nature Switzerland AG. Descargado de https://doi.org/10.1007/978-3-030-35746-7_8 doi: 10.1007/978-3-030-35746-7_8

Amazon. (2023). *Api restful*. https://aws.amazon.com/es/what-is/restful-api/.

Ambler, S. (2004). *The object primer 3/e*. Cambridge, UK: Cambridge.

Antel. (s.f.). *Elastic cloud antel*. https://minubeantel.uy/. (Último acceso: 26-09-2024)

Apple. (2024). *Healthkit — apple developer documentation*. Descargado de https://developer.apple.com/documentation/healthkit/about_the_healthkit_framework (Último acceso: 07-10-2024)

Araujo, D., Vigna, I., y Betarte, J. (2024). Finglix 2.0 - plataforma de dosificación de precisión informada por modelos. *Udelar. FI*.

Atlassian. (s.f.). *Metodología ágil*. https://www.atlassian.com/es/agile. (Último acceso: 26-09-2024)

Bruynseels, K., de Sio, F. S., y van den Hoven, J. (2018). Digital twins in health care: ethical implications of an emerging engineering paradigm. *Frontiers in Genetics*, *9*(31). Descargado de https://doi.org/10.3389/fgene.2018.00031 doi: 10.3389/fgene.2018.00031

Burgos, E. (2023). *Redux vs zustand, ¿cuál debo elegir?* https://www.linkedin.com/pulse/redux-vs-zustand-cu%C3%A1l-debo-elegir-esteban-burgos/. (Último acceso: 26-09-2024)

Business insider. (2023). https://www.businessinsider.com/.

Chart, L. (s.f.). *Lucid chart.* https://lucidchart.com/. (Último acceso: 26-09-2024)

de Salud Pública, M. (s.f.). *Historia clínica electrónica nacional.* https://www.gub.uy/ministerio-salud-publica/tramites-y-servicios/servicios/historia-clinica-electronica-nacional. (Último acceso: 26-09-2024)

Drocco, A., Facal, T., y Pirotto, A. (2021). Plataforma para dosificación de precisión informada por modelos. *Udelar. FI*.

ELEM Biotech SL. (2015). *Elem the virtual humans factory.* https://elem.bio/elem-upcoming-media.html. (Último acceso: 26-09-2024)

Exactcure. (2020). *Simulation-paracetamol – exactcure.* Descargado de https://www.exactcure.com/simulation-paracetamol/

Expressjs. (s.f.). https://expressjs.com/. (Último acceso: 26-09-2024)

fabriziosandri. (2022). *R-integration.* https://www.npmjs.com/package/r-integration. (Último acceso: 26-09-2024)

Fit bit flex. (2023). https://staticcs.fitbit.com/content/assets/help/manuals/manual_flex_2_es.pdf.

Flower, R. J. (2013). Pharmacology 2.0. *British Journal of Clinical Pharmacology*, *76*(5), 625–629. Descargado de https://doi.org/10.1111/bcp.12088 doi: 10.1111/bcp.12088

Foundation, T. E. (2024). *Running ditto • eclipse ditto™ • a digital twin framework.* Descargado de https://eclipse.dev/ditto/installation-running.html (Último acceso: 22-10-2024)

Foundation, T. L. (2024). *Kubernetes.* Descargado de https://kubernetes.io/es/ (Último acceso: 22-10-2024)

Friedmann, D. (s.f.). Estudios de caso de salud digital ediciÓn 02 implementación de la historia clínica electrónica nacional de uruguay. https://publications.iadb.org/publications/spanish/viewer/Implementacion-de-la-Historia-Clinica-Electronica-Nacional-de-Uruguay.pdf. (Último acceso: 26-09-2024)

Google. (s.f.). *Google drive.* https://drive.google.com/. (Último acceso: 26-09-2024)

Google. (2024). *Get started with health connect.* Descargado de https://developer.android.com/health-and-fitness/guides/health-connect/develop/get-started (Último acceso: 07-10-2024)

Grupo interdisciplinario en dosificación de precisión. (s.f.). https://gidp.ei.udelar.edu.uy/. (Último acceso: 20-09-2024)

Insider intelligence. (2023). https://www.linkedin.com/company/insider-intelligence/.

Json web token. (s.f.). https://jwt.io/. (Último acceso: 26-09-2024)

Kariya, Y., y Honma, M. (2024). Applications of model simulation in pharmacological fields and the problems of theoretical reliability. *Drug Meta-*

bolism and Pharmacokinetics, *56*, 100996. Descargado de https://www.sciencedirect.com/science/article/pii/S1347436724000028 doi: https://doi.org/10.1016/j.dmpk.2024.100996

Kluwe, F., Michelet, R., Mueller-Schoell, A., Maier, C., Klopp-Schulze, L., van Dyk, M., ... Kloft, C. (2021). Perspectives on model informed precision dosing in the digital health era: Challenges, opportunities, and recommendations. *Clinical pharmacology And therapeutics*, *109*(1). www.cpt-journal.com.

Kollewe, J. (2021, 17 de September). Gsk teams with king's college to use ai to fight cancer. *The Guardian*. Descargado de https://www.theguardian.com/business/2021/sep/17/gsk-teams-with-kings-college-to-use-ai-to-fight-cancer (Último acceso: 26-09-2024)

Lehner, D., Pfeiffer, J., Tinsel, E., Strljic, M. M., Sint, S., Vierhauser, M., ... Wimmer, M. (2022). Digital twin platforms: Requirements, capabilities, and future prospects. *IEEE Software*, 53-61. doi: 10.1109/MS.2021.3133795

Liu, Y., Li, J., Xiao, S., Liu, Y., Bai, M., Gong, L., ... Chen, D. (2023). Revolutionizing precision medicine: Exploring wearable sensors for therapeutic drug monitoring and personalized therapy. *Biosensors*, *13*(7), 726. Descargado de https://doi.org/10.3390/bios13070726 doi: 10.3390/bios13070726

Mattermost. (s.f.). *Mattermost.* https://mattermost.com/. (Último acceso: 26-09-2024)

Meta. (s.f.). *Whatsapp.* https://whatsapp.com/. (Último acceso: 26-09-2024)

Meta. (2024). *Introducing hooks.* Descargado de https://legacy.reactjs.org/docs/hooks-intro.html (CÚltimo acceso: 7-12-2024)

Microsoft. (s.f.). *Github.* https://github.com/. (Último acceso: 26-09-2024)

Microsoft. (s.f.). *Vscode.* https://code.visualstudio.com/. (Último acceso: 26-09-2024)

Microsoft. (2024). *Typescript.* Descargado de https://www.typescriptlang.org/ (CÚltimo acceso: 7-12-2024)

MQTT. (2022). *Mqtt: The standard for iot messaging.* Descargado de https://mqtt.org/ (Último acceso: 09-10-2024)

Mrgsolve. (s.f.). *Mrgsolve.* Descargado de https://mrgsolve.org/ (Último acceso: 22-10-2024)

Nginx. (2023). https://nginx.org/en/docs/.

NodeJS. (2024). *Nodejs.* Descargado de https://nodejs.org/ (CÚltimo acceso: 7-12-2024)

OMS. (2005). *Constitution of the world health organization: Principles.*

Overleaf. (s.f.). *Overleaf.* https://overleaf.com/. (Último acceso: 26-09-2024)

Postgresql. (s.f.). https://www.postgresql.org/. (Último acceso: 26-09-2024)

Powers, R., Etezadi-Amoli, M., Arnold, E. M., Kianian, S., Mance, I., Gibiansky, M., ... Ullal, A. V. (2021). Smartwatch inertial sensors continuously mo-

nitor real-world motor fluctuations in parkinson's disease. *Science Translational Medicine*, *13*(579). doi: 10.1126/scitranslmed.abd7865

Query, R. (2024). *Rtk query.* https://redux-toolkit.js.org/rtk-query/overview. (Último acceso: 26-09-2024)

Rascia, T. (2024). *Crud rest api with node.js, express, and postgresql.* https://blog.logrocket.com/crud-rest-api-node-js-express-postgresql/. (Último acceso: 26-09-2024)

ReactJS. (s.f.). *Reactjs.* https://react.dev/. (Último acceso: 26-09-2024)

React router. (s.f.). https://reactrouter.com/. (Último acceso: 26-09-2024)

Salud uy. (s.f.). https://centrodeconocimiento.agesic.gub.uy/web/salud.uy. (Último acceso: 26-09-2024)

Sequelize. (s.f.). https://sequelize.org/. (Último acceso: 26-09-2024)

Sharma, I. (2024). *Vite vs create-react-app: A detailed comparison.* https://www.tatvasoft.com/outsourcing/2024/07/vite-vs-create-react-app.html. (Último acceso: 26-09-2024)

Three.js. (s.f.). https://threejs.org/. (Último acceso: 26-09-2024)

Virtuozzo. (2023). https://www.virtuozzo.com/.

Visily. (s.f.). *Visily.* https://visily.ai/. (Último acceso: 26-09-2024)

Y. Wurmser. (2023). *Wearable tech in healthcare: Smart medical devices and trends in 2023.* https://www.emarketer.com/content/wearables-2023/.

ZeppHealth. (2020). *Huami web api.* Descargado de https://github.com/zepp-health/rest-api/wiki

Zoom. (s.f.). *Zoom.* https://zoom.com/. (Último acceso: 26-09-2024)

Anexo A

Anexo Manuales de usuario

En esta sección se detallan las funcionalidades de la plataforma, describiéndolas paso a paso para cada flujo.

A.1. Sesión

Este flujo consiste en autenticar usuarios desde la pantalla de inicio de sesión (figura A.1 con la contraseña correspondiente para ese médico.

También se persiste la sesión, es decir, que al recargar la página o cerrar el navegador la sesión se mantiene. Se cuenta con *refresh* del token, para que a pesar de que este expire, se pueda pedir uno nuevo y el usuario no se entere.

Finalmente, se cuenta con cerrado de sesión, que elimina todos los datos del usuario guardados en el navegador e invalida el *token* del mismo.

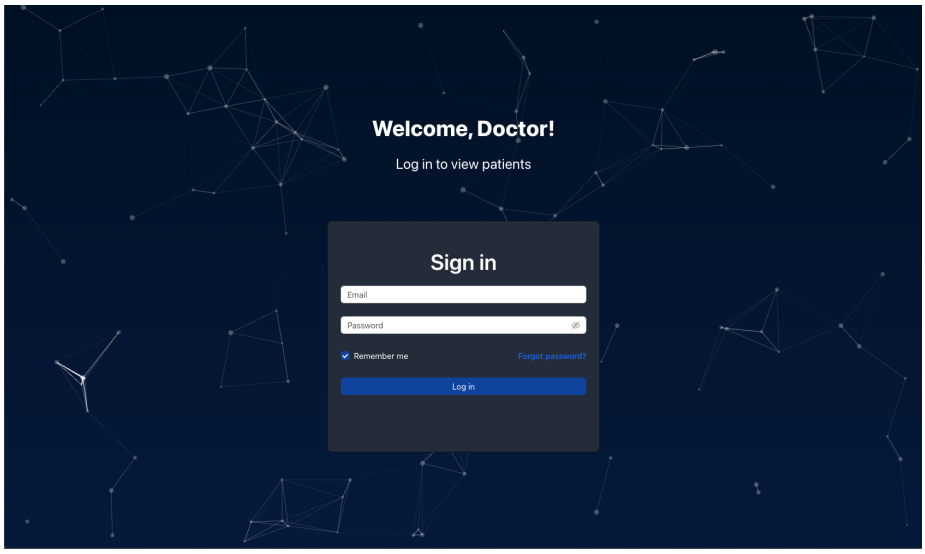


Figura A.1: Inicio de sesión

A.2. Visualización de paciente

Este flujo consiste en la visualización de un paciente. Para lograr esto, se deben seguir los siguientes pasos:

- Luego de iniciar sesión se presenta una lista con los diferentes pacientes del médico, como se muestra en la figura A.2. Estos pacientes son los que efectivamente están asignados a ese usuario y no todos los que existen en el sistema.
- De manera opcional, se puede filtrar a los pacientes tanto por nombre como por cédula de identidad. Se selecciona un paciente de la lista (figura A.3)
- Esta selección navega hacia la página principal del paciente. En esta página se puede apreciar una disposición de distintos elementos (figura A.4). A la izquierda se encuentra el modelo de datos en vivo del paciente. Estos son los datos obtenidos mediante los *wearables*. Si se desea consultar alguno de estos datos, basta con colocar el cursor sobre el ícono correspondiente. Esto mostrará el último dato tomado con su unidad respectiva, así como también la fecha y hora en la que se tomó esta medida A.5. En el medio de la pantalla se puede ver la lista de datos del paciente que fueron ingresados tanto de forma manual como guardados al realizar simulaciones, así como también un botón para agregar datos y un indicador de que el paciente

tiene un dispositivo móvil vinculado. A la derecha se puede visualizar una lista que muestra los tratamientos activos del paciente.

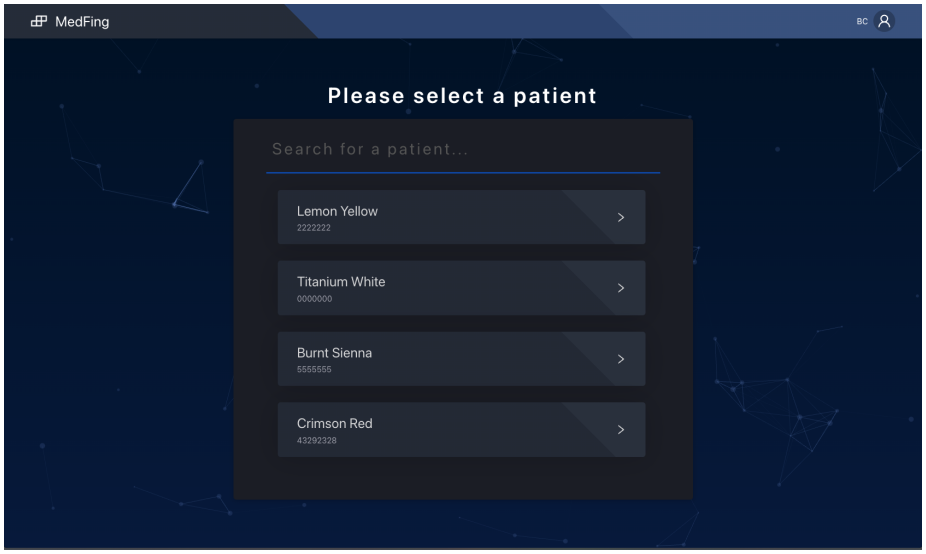


Figura A.2: Visualización de paciente

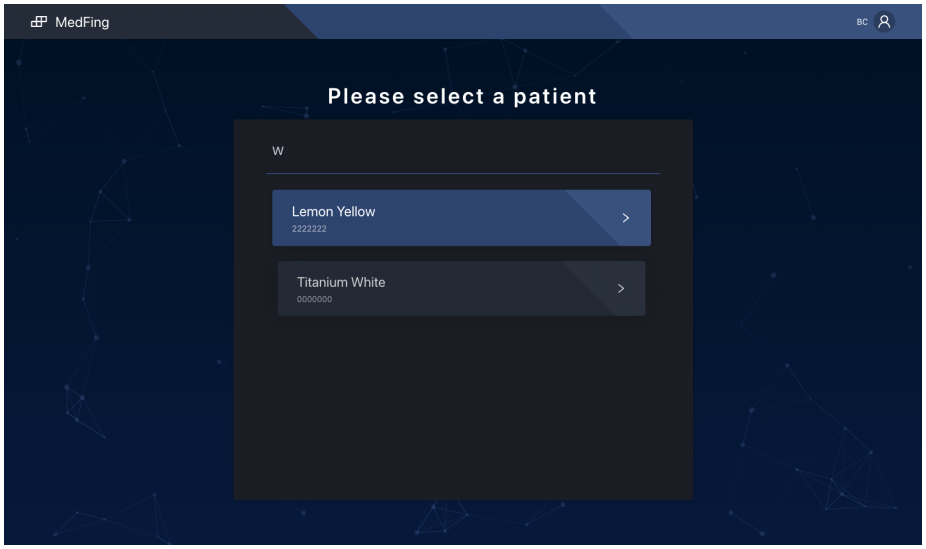


Figura A.3: Visualización de paciente

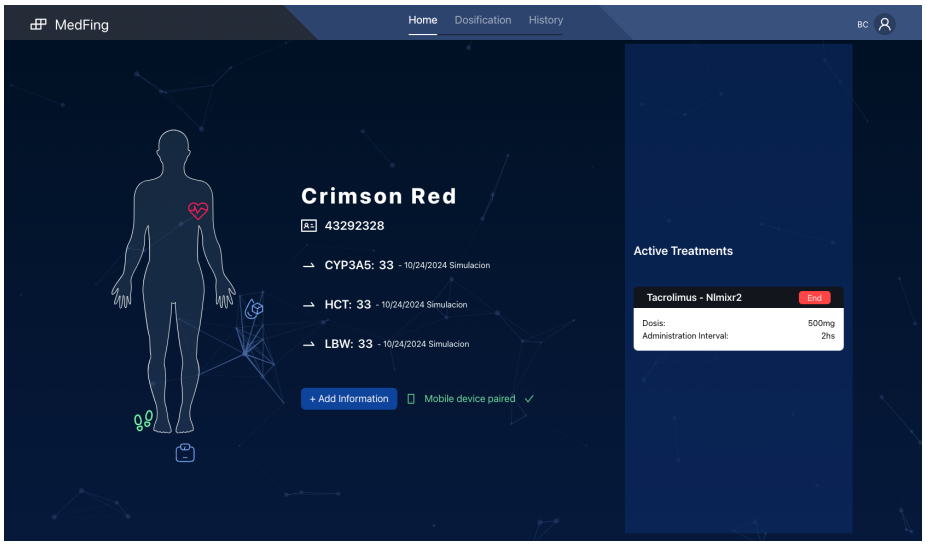


Figura A.4: Visualización de paciente

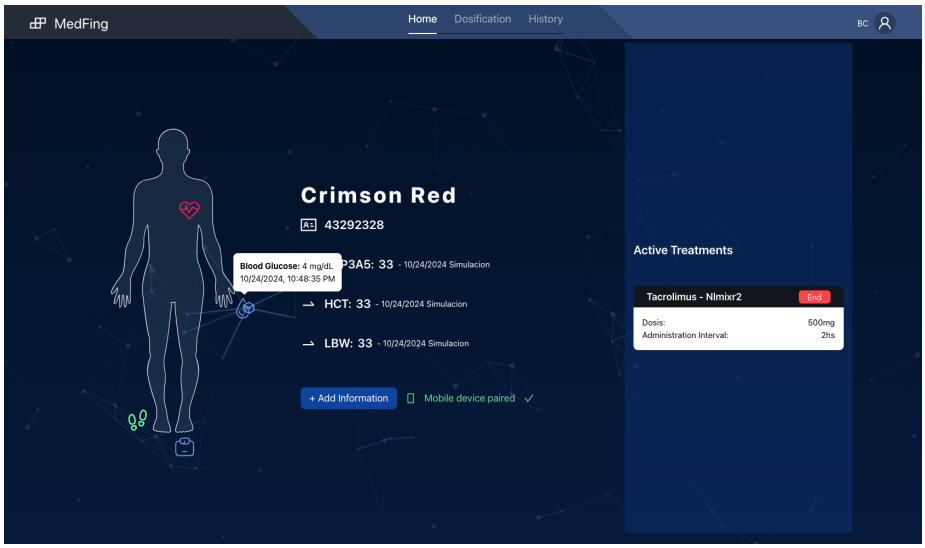


Figura A.5: Visualización de paciente

A.3. Ingresar datos manuales

Este flujo consiste en el ingreso de datos manuales, siguiendo estos pasos:

- En primer lugar, se debe presionar el botón mostrado en la figura A.6 para agregar información.
- Se puede elegir agregar un parámetro (figura A.7) o un comentario (figura A.8). Para agregar un parámetro se debe seleccionar primero el tipo de parámetro y luego ingresar el valor. En el caso del comentario, se ingresa en un recuadro de texto libre.
- Para guardar cualquiera de estos dos tipos de información se debe presionar el botón de *Add* y finalmente verificar en la página principal del usuario que se hayan agregado correctamente, como se muestra en la figura A.9. También se puede visualizar todo el histórico de los datos navegando hasta la pestaña de historia (figura A.10)

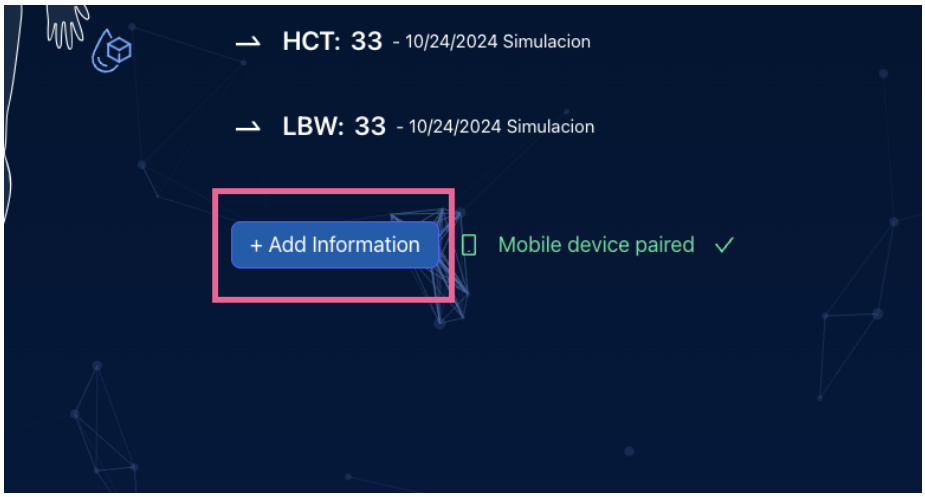


Figura A.6: Ingresar datos manuales

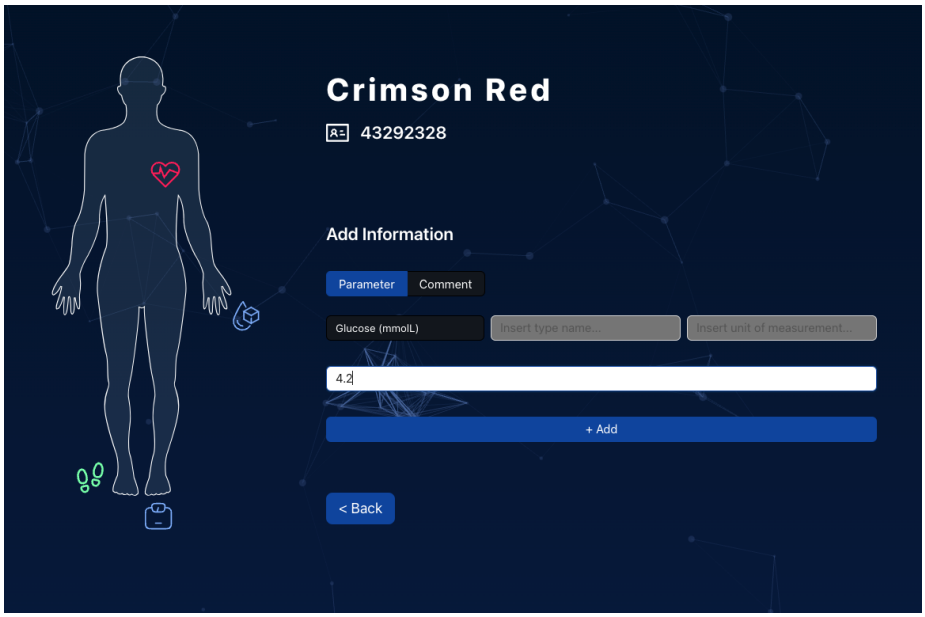


Figura A.7: Ingresar datos manuales

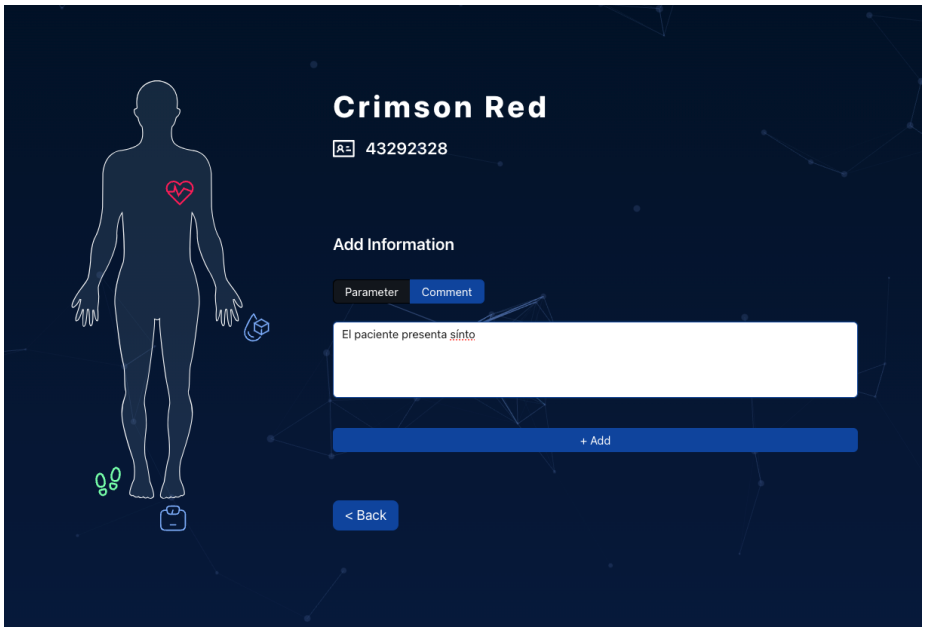


Figura A.8: Ingresar datos manuales

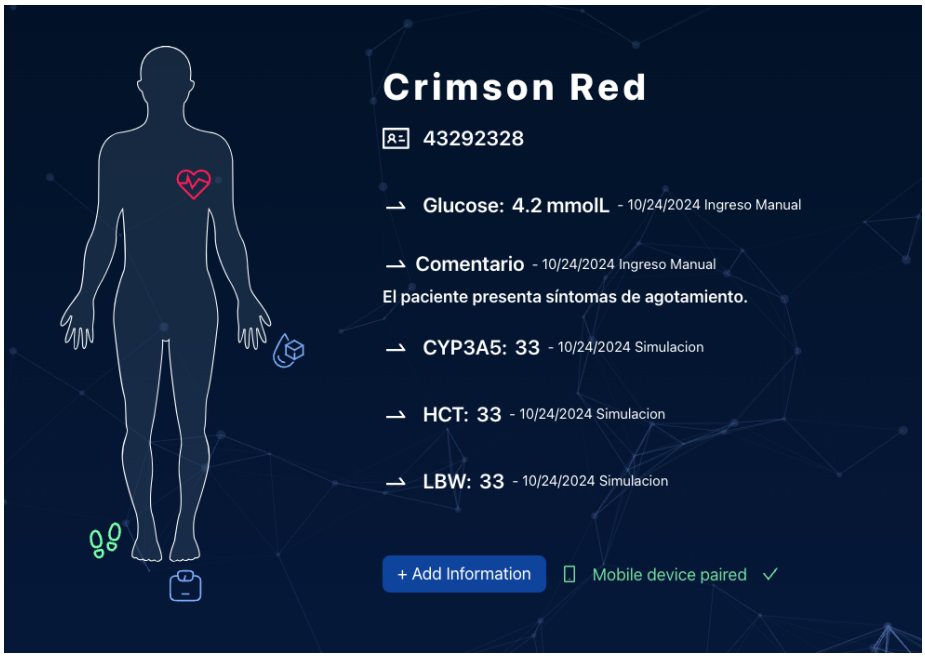


Figura A.9: Ingresar datos manuales

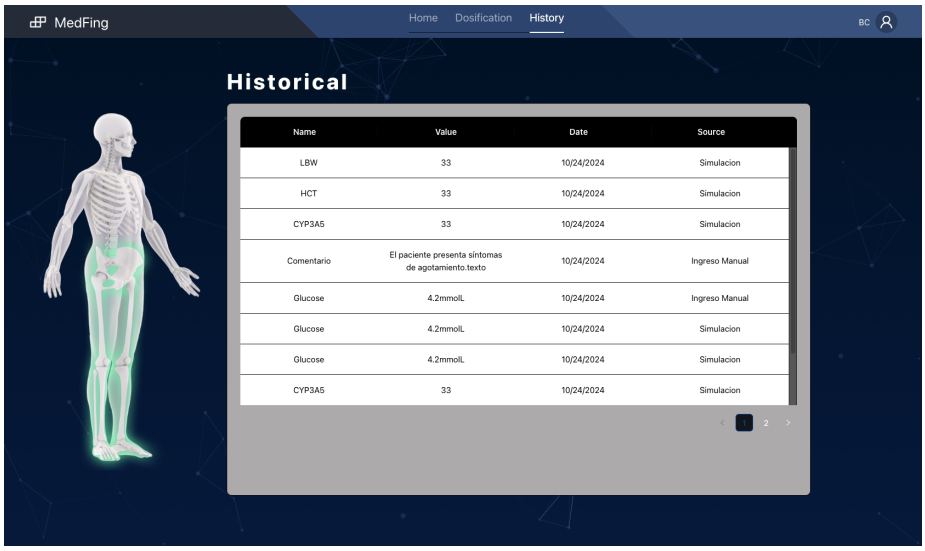


Figura A.10: Ingresar datos manuales

A.4. Generar tratamiento

En esta sección se explica cómo generar un tratamiento con la simulación o simulaciones deseadas:

- En primer lugar, se debe navegar hasta la pestaña de dosificación que se encuentra en el encabezado como se muestra en A.11.
- Luego, se debe seleccionar un proveedor de simulación (figura A.12). En esta iteración del proyecto solo se incluyen Mrgsolve y Finglix. Para este ejemplo se selecciona Mrgsolve.
- Se debe seleccionar una droga de las opciones disponibles: Rifampicina, Azithro, Meropenem e Insulina. Se selecciona como ejemplo Insulina y se completan los parámetros requeridos en el formulario mostrado en A.13, y se presiona el botón de la esquina superior derecha *Simulate* para simular.
- En la figura A.14 se puede apreciar la simulación recién generada. En este punto se puede elegir iniciar el tratamiento, guardarlo como borrador o agregar más simulaciones para comparar.
- Si se deciden agregar más simulaciones, se debe presionar el botón *Add simulation*, acción que mostrará un modal en el que se pueden ingresar los datos de la nueva simulación (figura A.15).
- Luego de simular por segunda vez, ambas gráficas se muestran en el mismo *canvas* con el fin de que el médico las pueda comparar, como se puede apreciar en A.16.
- Si en lugar de Mrgsolve se elige Finglix como proveedor, el procedimiento y los resultados son similares. En el título del tratamiento se indica con qué proveedor se realizan las simulaciones (figura A.17)
- Para seleccionar la dosis efectiva que se le va a proporcionar al paciente, o para revisar la información de otra simulación, basta simplemente con hacer *click* en la gráfica deseada (figura A.18). Esta acción tiene consecuencias en la interfaz de usuario, como cambiar la intensidad del color de las venas del modelo humano de la izquierda para representar la comparación entre las distintas opciones o modificar los colores de las tablas para que sea más claro cual es la simulación seleccionada.
- Para iniciar el tratamiento se debe presionar el botón *Start Treatment* que se muestra en la figura A.19. Esta acción inicia el un tratamiento con la dosis en base a la última simulación seleccionada. Esta dosis no se puede cambiar por diseño: para mantener trazabilidad e historia de los datos, una vez que se inicia un tratamiento no se puede modificar. Si el médico desea modificar la dosis seleccionada, es necesario que cree un tratamiento nuevo.

- Si no se desea seleccionar la dosis al momento de realizar las simulaciones, existe la opción de guardar como borrador. Para eso se debe presionar el botón *Save as Draft*, lo que creará el tratamiento de forma que se pueda seleccionar la dosis más tarde.

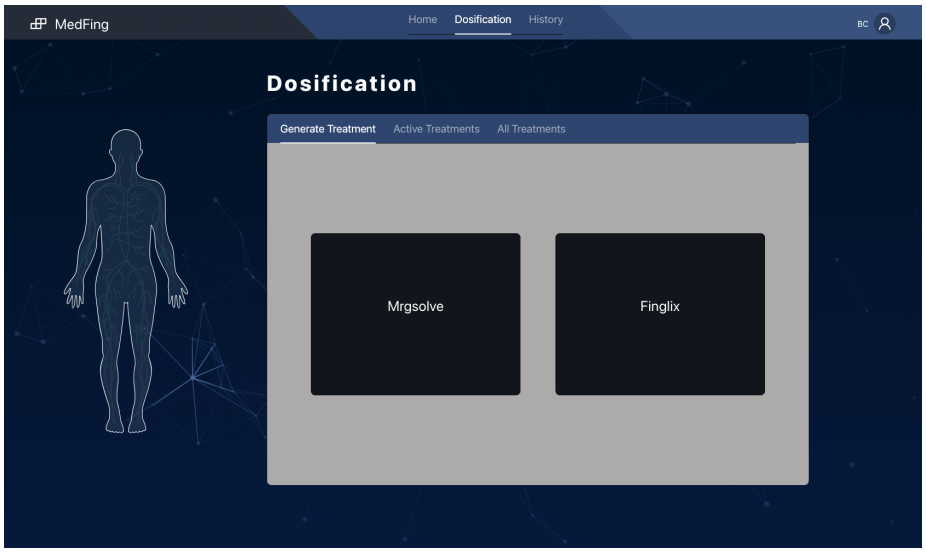


Figura A.11: Generar tratamiento

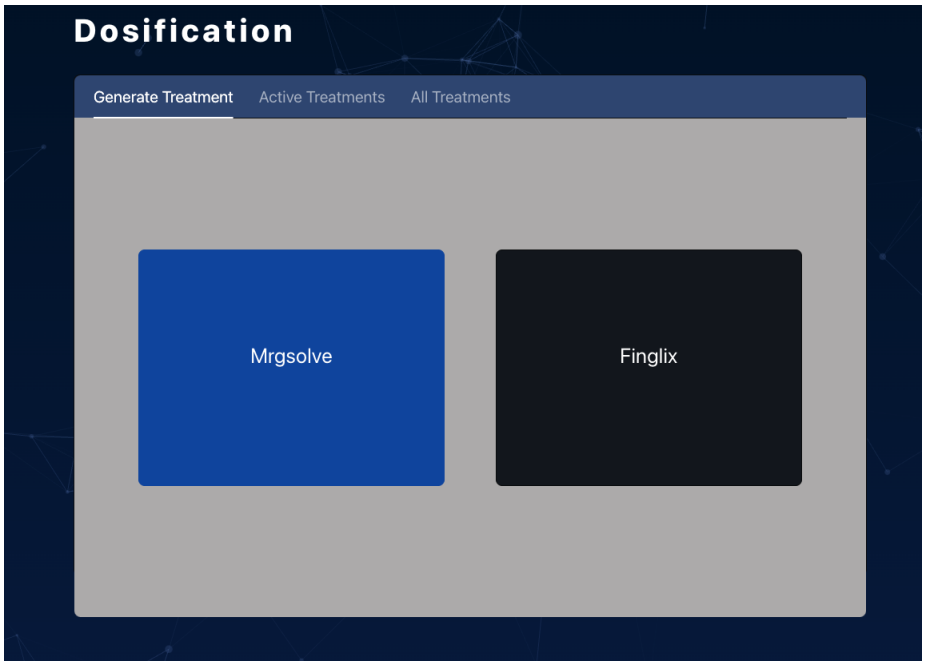


Figura A.12: Generar tratamiento

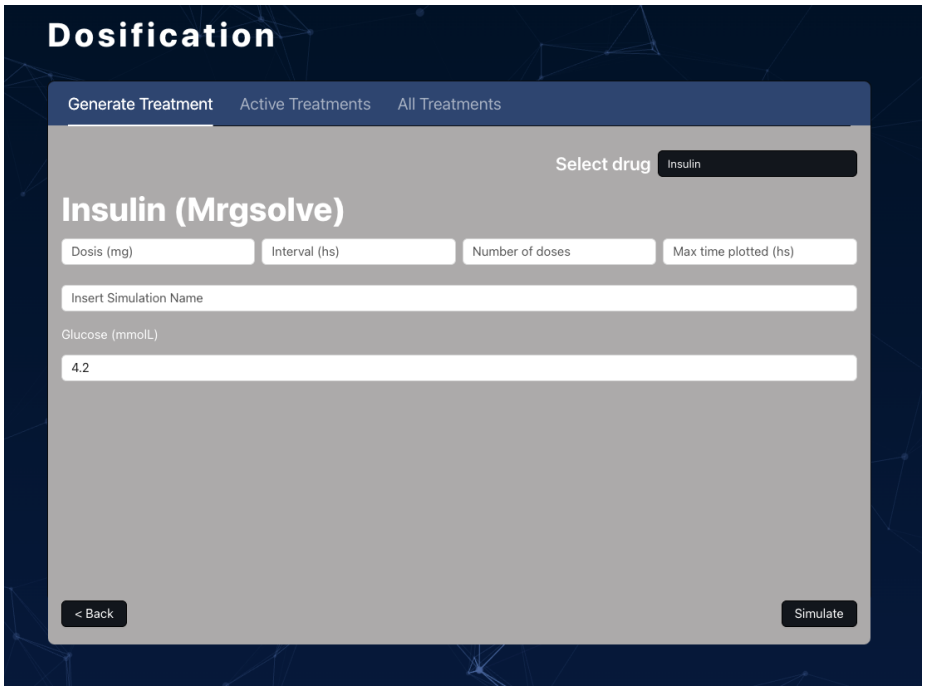


Figura A.13: Generar tratamiento

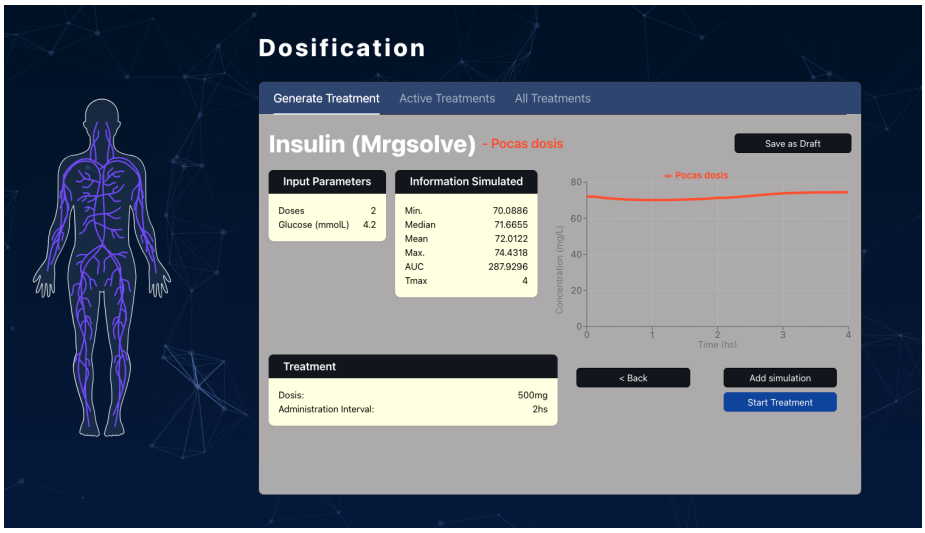


Figura A.14: Generar tratamiento

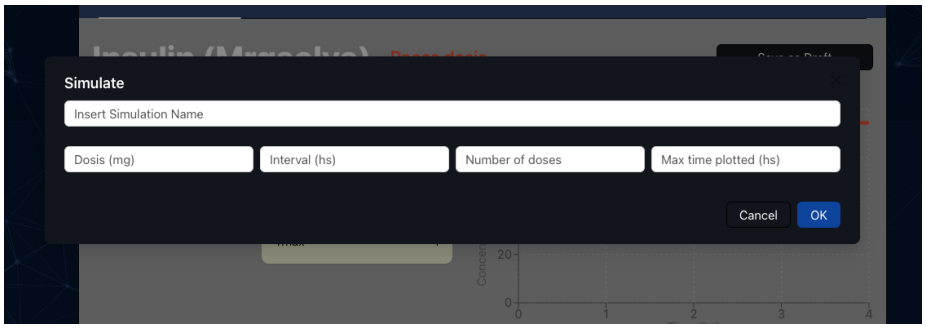


Figura A.15: Generar tratamiento

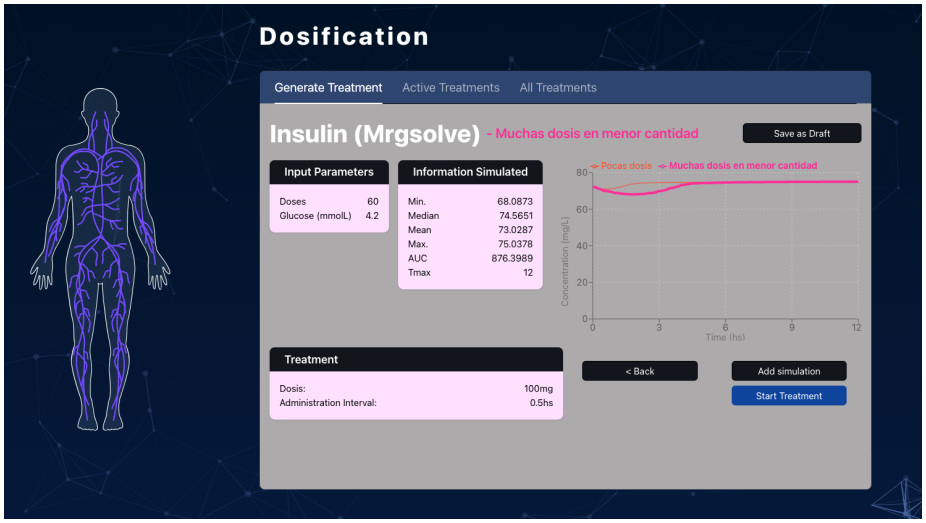


Figura A.16: Generar tratamiento

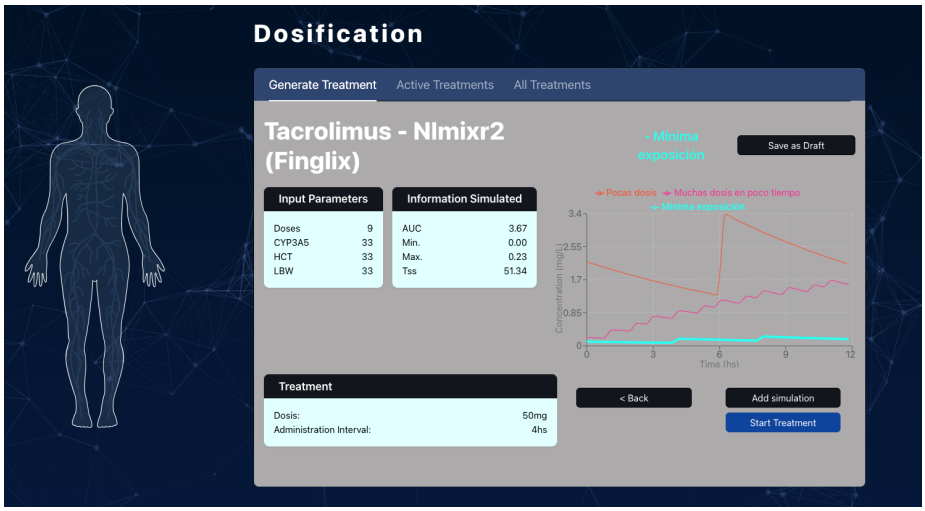


Figura A.17: Generar tratamiento

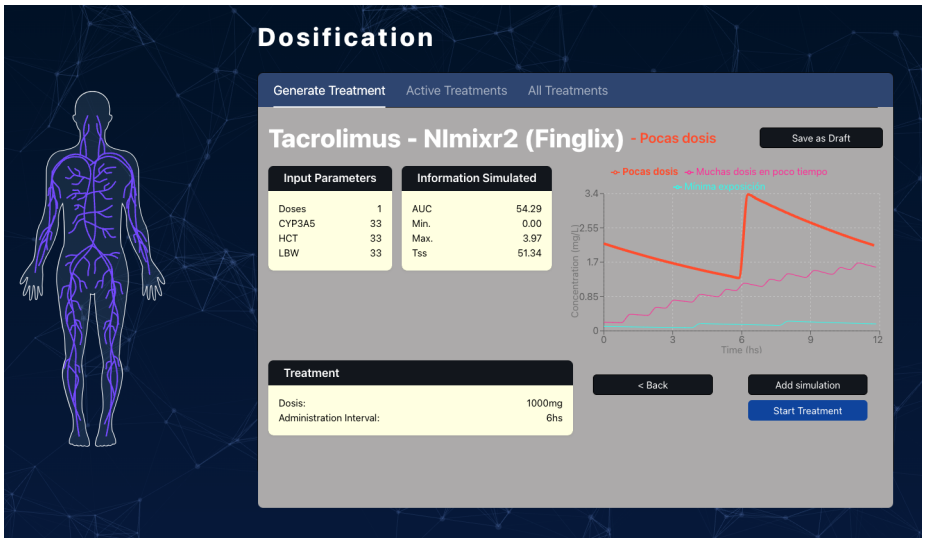


Figura A.18: Generar tratamiento

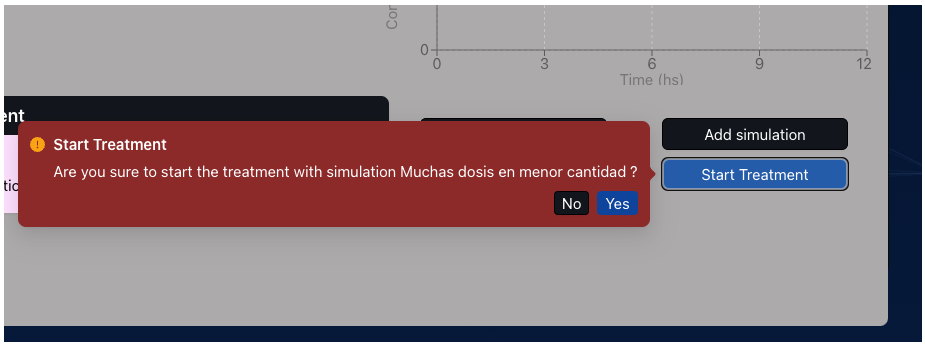


Figura A.19: Generar tratamiento

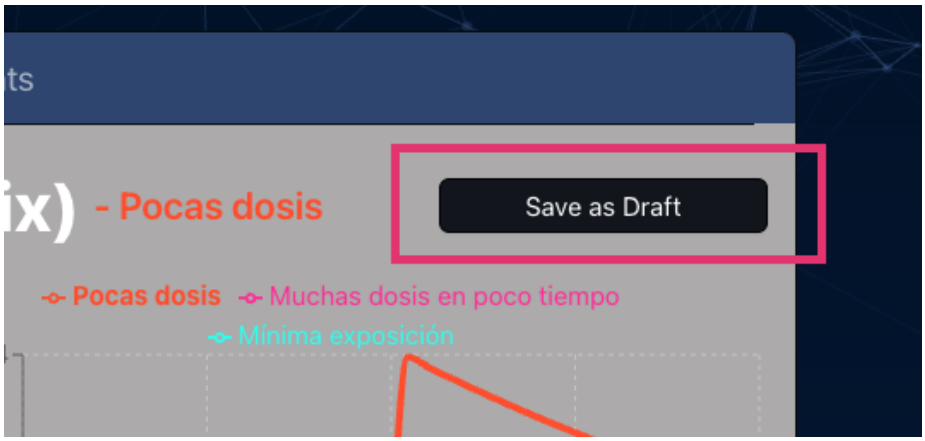


Figura A.20: Generar tratamiento

A.5. Visualizar tratamiento

Para visualizar un tratamiento existente se siguen estos pasos:

- En el recuadro principal de la pestaña de dosificación se selecciona la pestaña *Active Treatments*. Esto mostrará una grilla con los diferentes tratamientos activos del usuario, como se puede ver en la figura A.21.
- Si se desea acceder a la lista de todos los tratamientos, se selecciona la pestaña *All Treatments* del recuadro principal de dosificación. En esta lista se encuentran todas las entidades de tipo tratamiento creadas para este paciente, independientemente del estado que tengan A.22.

- Al seleccionar un tratamiento, se accede a la vista detallada del mismo. Esta vista cuenta con información sobre las simulaciones y es similar a lo que se puede visualizar cuando se está creando el tratamiento (figura A.23).
- Para terminar un tratamiento, se puede hacer tanto desde el botón rojo *End treatment* de la vista principal (figura A.23) como desde el botón *End* de la grilla de la pestaña de *Active Treatments* (figura A.24).
- Si se está visualizando un tratamiento que está en estado borrador, se pueden añadir simulaciones o iniciar el tratamiento utilizando la sección de botones señalada en la figura A.25.

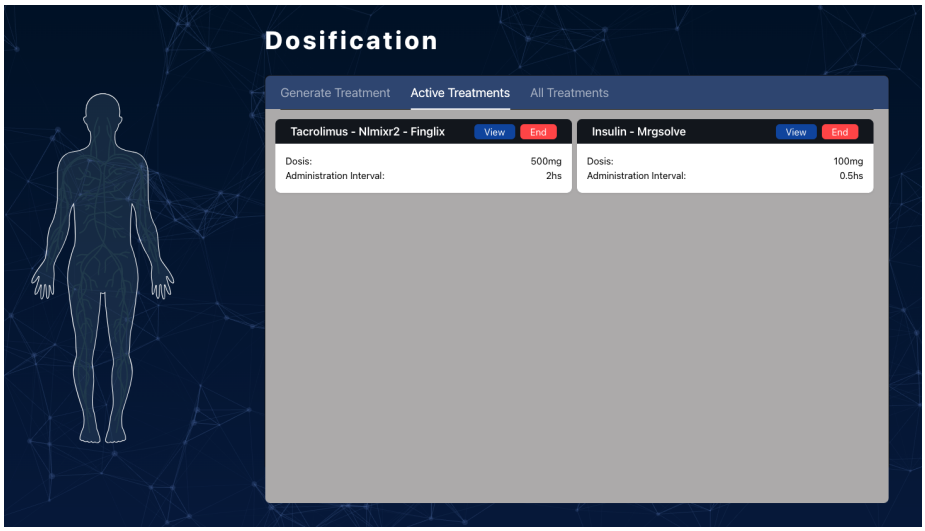


Figura A.21: Visualizar tratamiento

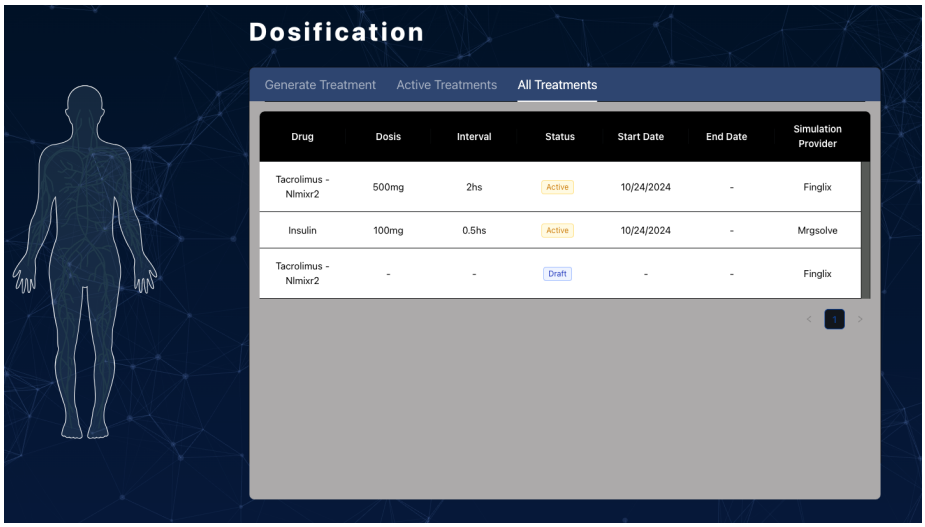


Figura A.22: Visualizar tratamiento

Figura A.23: Visualizar tratamiento

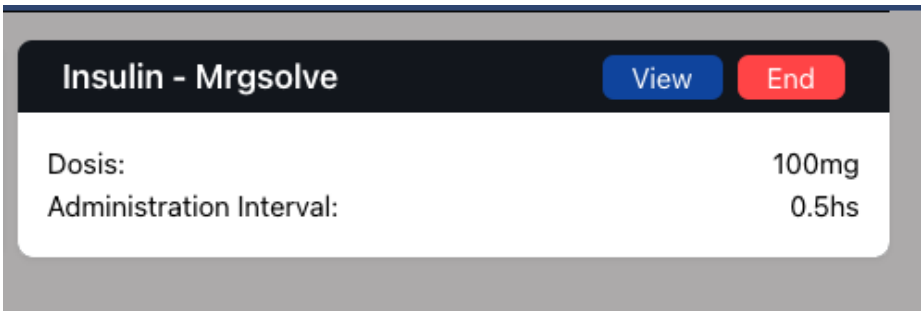


Figura A.24: Visualizar tratamiento

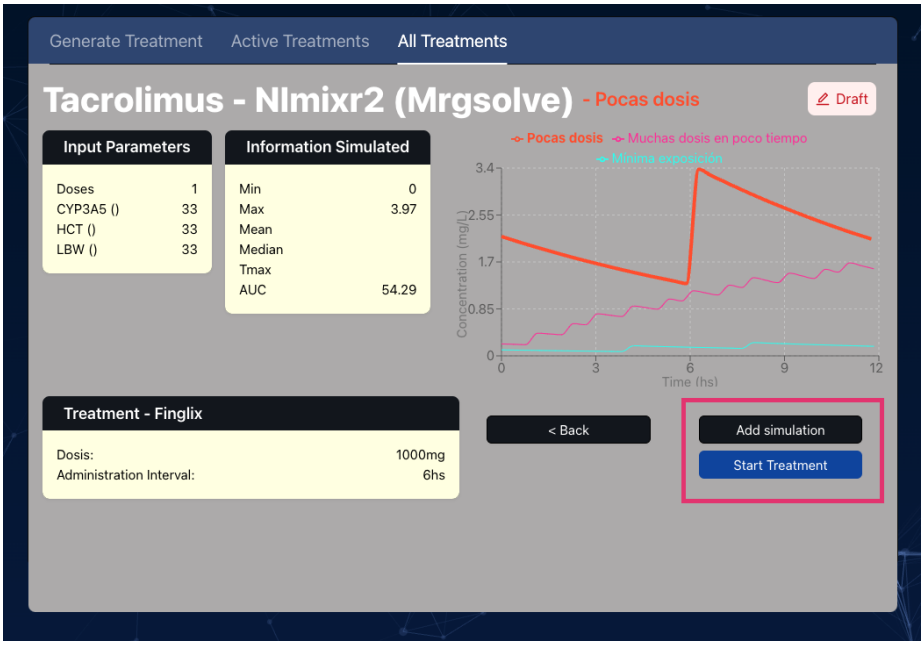


Figura A.25: Visualizar tratamiento

A.6. OTPs

Para realizar la vinculación de un dispositivo móvil con la aplicación web se deben seguir estos pasos:

- Durante la consulta, el médico solicita el emparejamiento desde la aplicación web (figura A.26), lo que genera un código OTP de cuatro dígitos

único para ese paciente y sesión. Este código es visible para el médico (figura A.27).

- El médico entrega el código OTP al paciente junto con la cédula de identidad, permitiendo el emparejamiento entre el perfil del paciente en el sistema y la aplicación móvil instalada en su dispositivo (figura A.28 y A.29).
- El paciente ingresa el OTP en la aplicación móvil, que lo envía al sistema para validarlo. El sistema verifica que el código coincida con el generado previamente, completando el proceso de verificación. Se notifica tanto al médico como al paciente sobre el éxito del emparejamiento (figura A.31 y figura A.30).

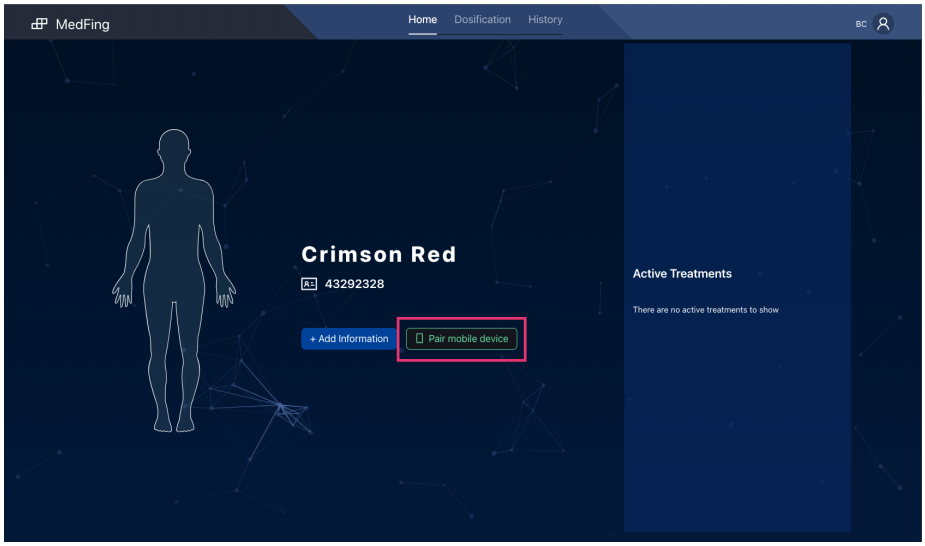


Figura A.26: Solicitud de OTP

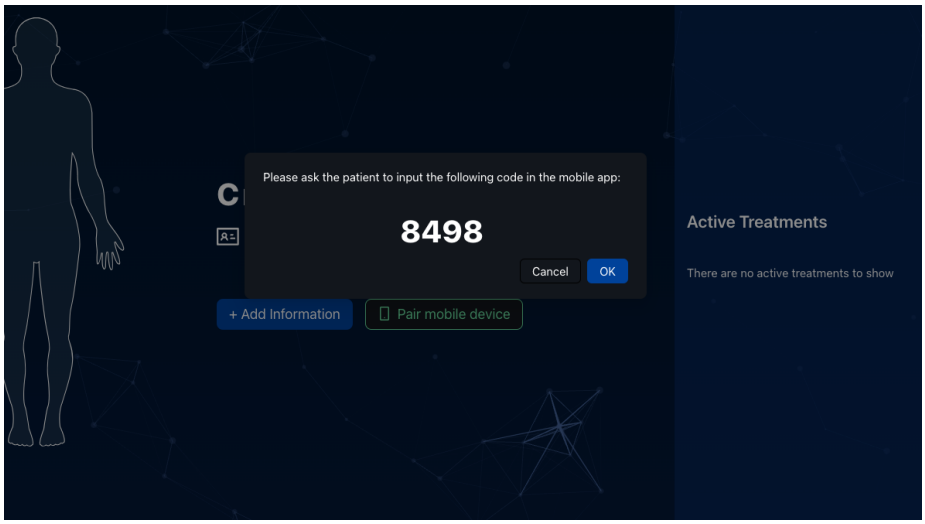


Figura A.27: Visualización del OTP

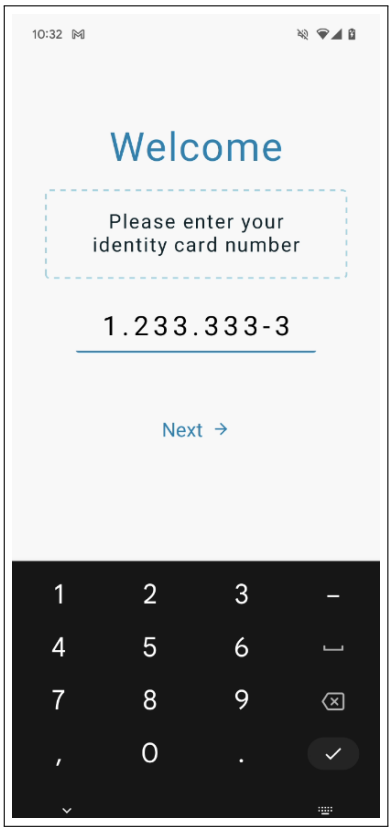


Figura A.28: Ingreso de CI

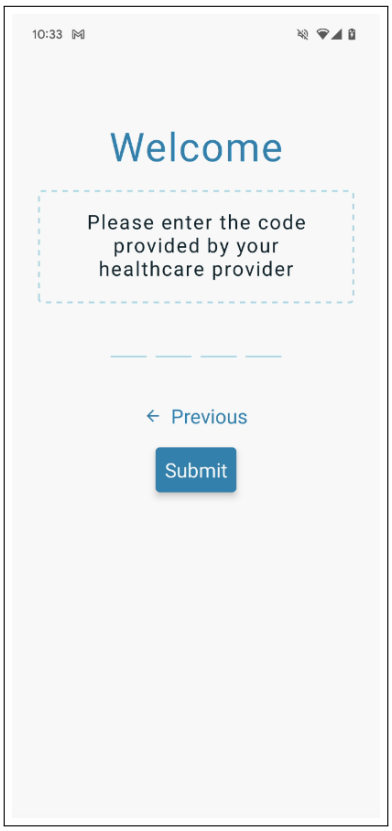


Figura A.29: Ingreso de OTP

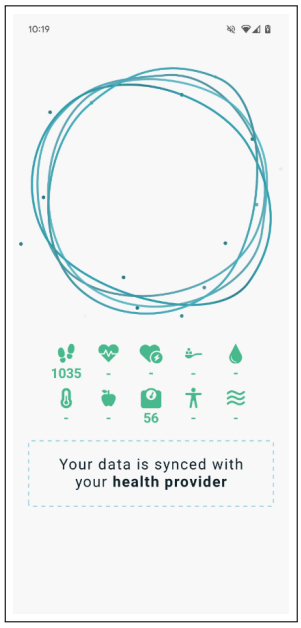


Figura A.30: Verificación en la aplicación móvil

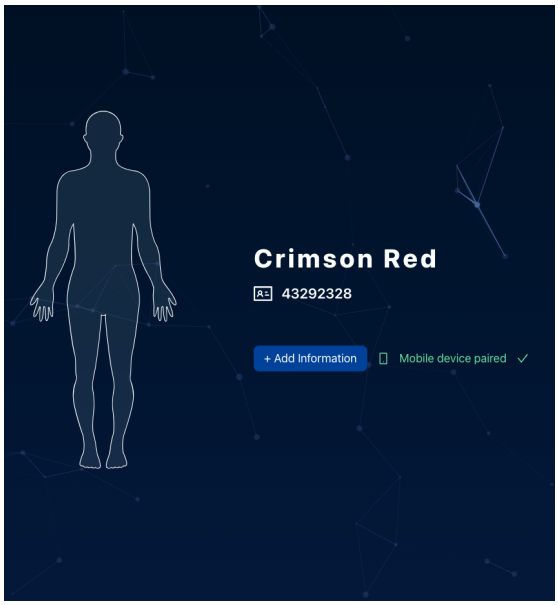


Figura A.31: Verificación en la web

A.7. Enviar datos desde la aplicación móvil

Para probar el flujo de datos entre la aplicación móvil y la aplicación web, se pueden seguir estos pasos para simular una medida de un dato de salud:

- Usando la aplicación Google Fit, buscar el tipo de dato que se desee probar, en este caso*Blood glucose*, como se muestra en la figura A.32.
- Agregar un valor en el campo correspondiente de la manera que se indica en la figura A.33.
- Para corroborar que el dato se esté registrando en la aplicación móvil, abrir dicha aplicación y verificar el valor correspondiente como se puede ver en la figura A.34. Se puede presionar cualquier ícono para leer su nombre en caso de que el ícono no sea lo suficientemente claro.
- Finalmente, para verificar el flujo de datos mediante MQTT con Ditto, el backend y la aplicación web, basta con ingresar a la pantalla principal del paciente para el que fueron tomados los datos y ubicar el cursor sobre el ícono que tiene el tipo de dato que se desea leer.

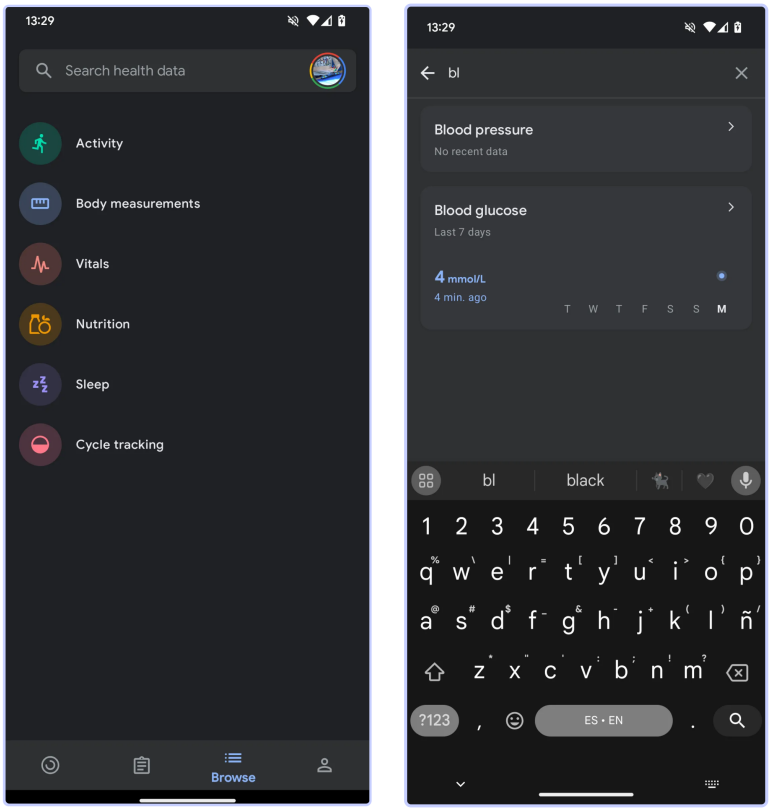


Figura A.32: Enviar datos desde la aplicación móvil

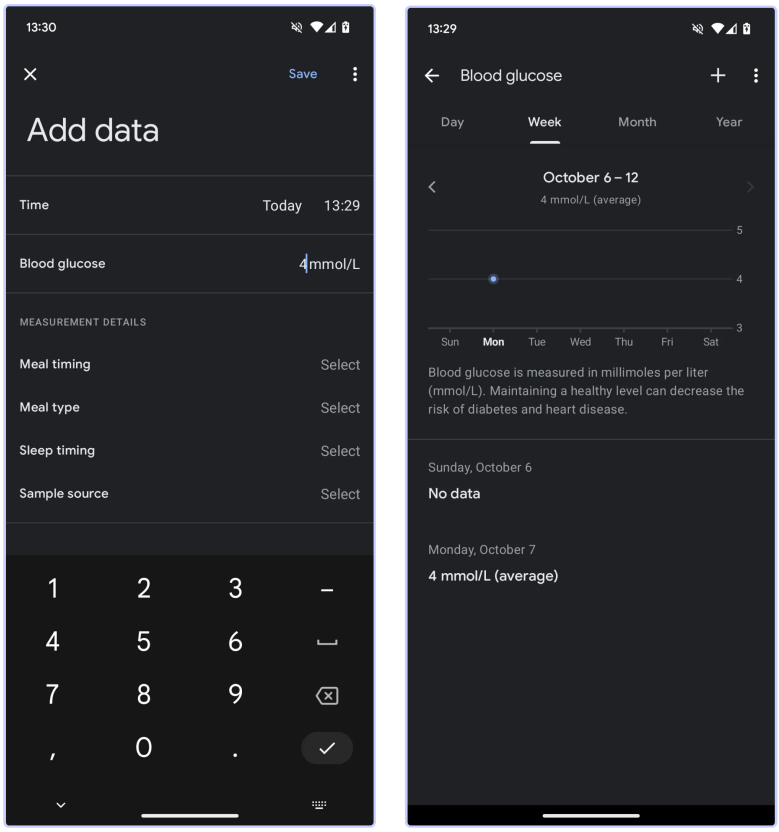


Figura A.33: Enviar datos desde la aplicación móvil

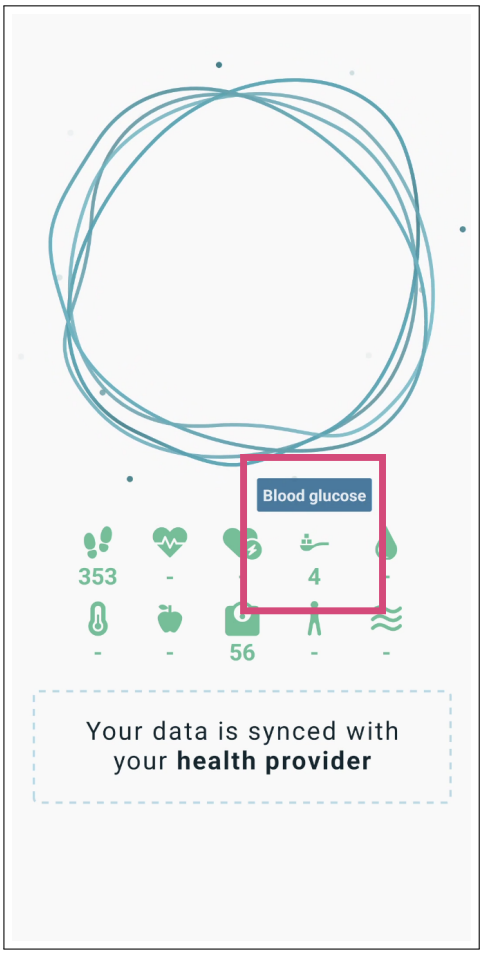


Figura A.34: Enviar datos desde la aplicación móvil

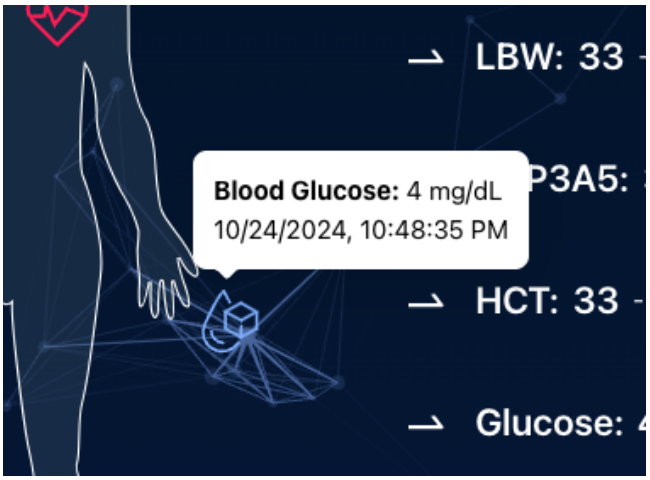


Figura A.35: Enviar datos desde la aplicación móvil

Anexo B

Anexo Validación

Este anexo cuenta con todos los resultados de la encuesta a los potenciales usuarios.

B.1. Contexto

B.1.1. Cargo/Profesión

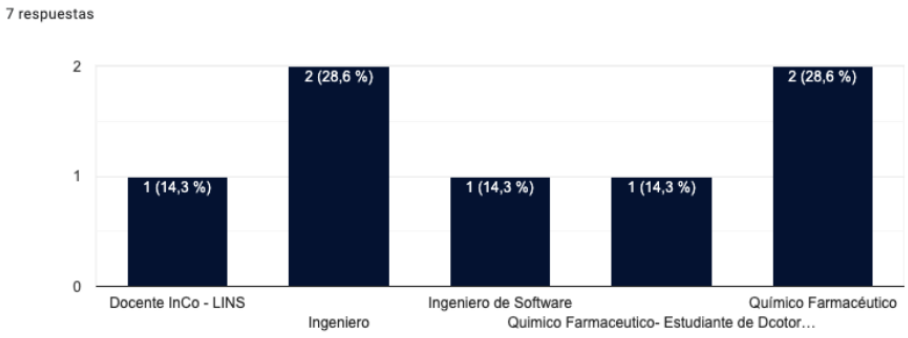


Figura B.1: Cargo/Profesión

B.1.2. ¿Está relacionado(a) con la medicina de precisión o tecnología de la salud?

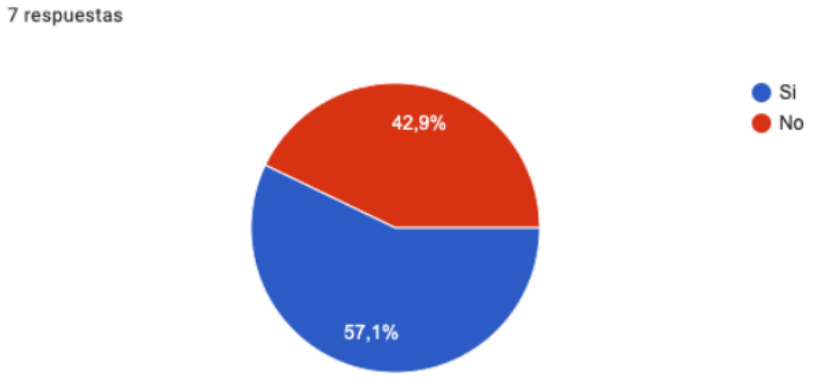


Figura B.2: ¿Está relacionado(a) con la medicina de precisión o tecnología de la salud?

B.1.3. ¿Es usted un futuro (o posible) usuario de la plataforma presentada?

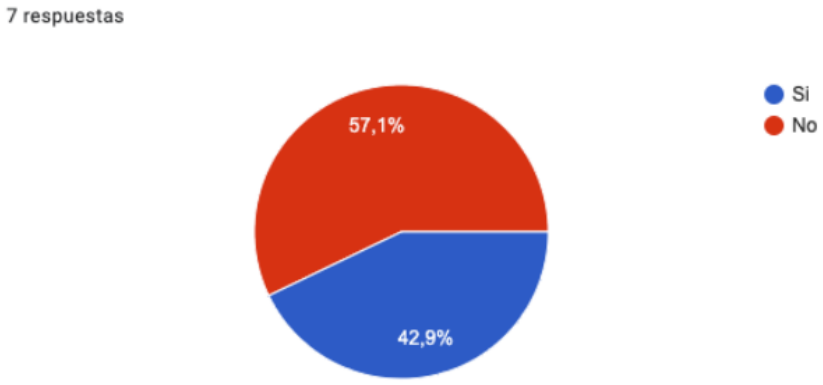


Figura B.3: ¿Es usted un futuro (o posible) usuario de la plataforma presentada?

B.1.4. ¿Conocía previamente el concepto de gemelos digitales?

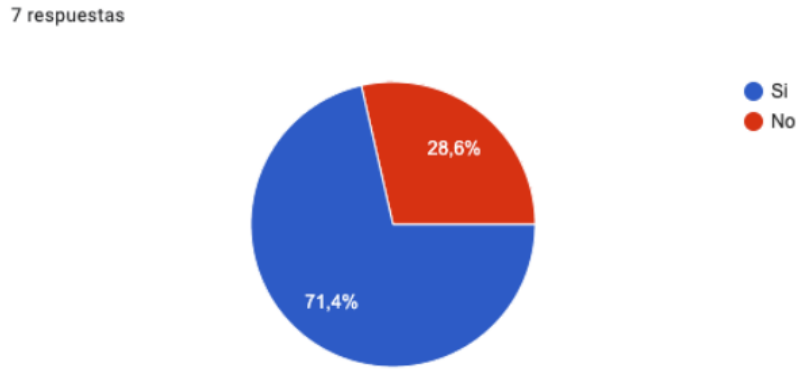


Figura B.4: ¿Ha utilizado alguna vez herramientas para simulaciones de dosis de medicamentos?

B.2. Objetivos

B.2.1. En su opinión, ¿cuán importante considera la personalización en los tratamientos médicos?

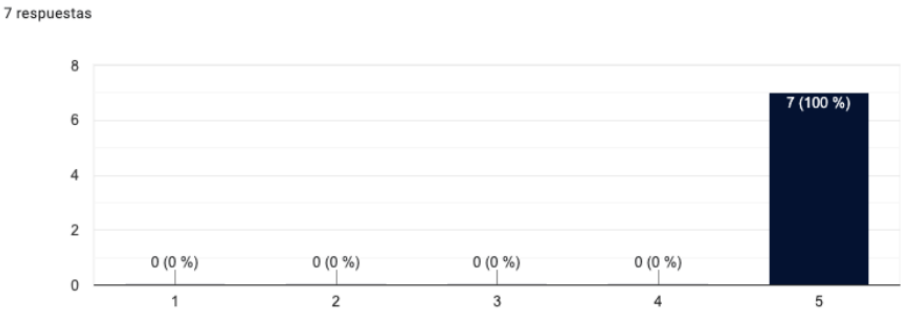


Figura B.5: En su opinión, ¿cuán importante considera la personalización en los tratamientos médicos?

B.2.2. ¿Cree que los gemelos digitales podrían mejorar la dosificación de precisión?

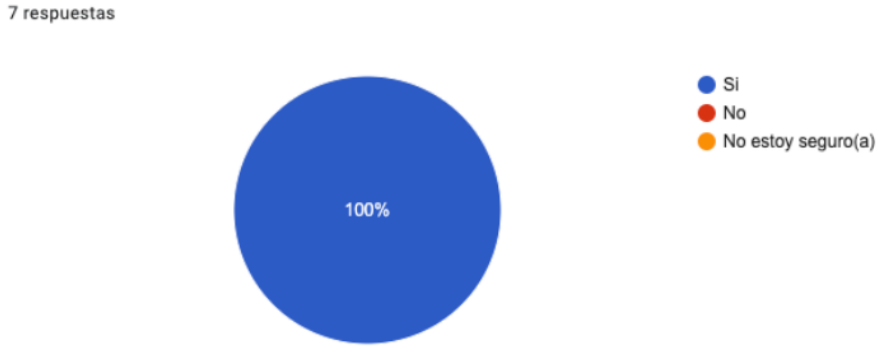


Figura B.6: ¿Cree que los gemelos digitales podrían mejorar la dosificación de precisión?

B.2.3. ¿Ha utilizado alguna vez herramientas para simulaciones de dosis de medicamentos?

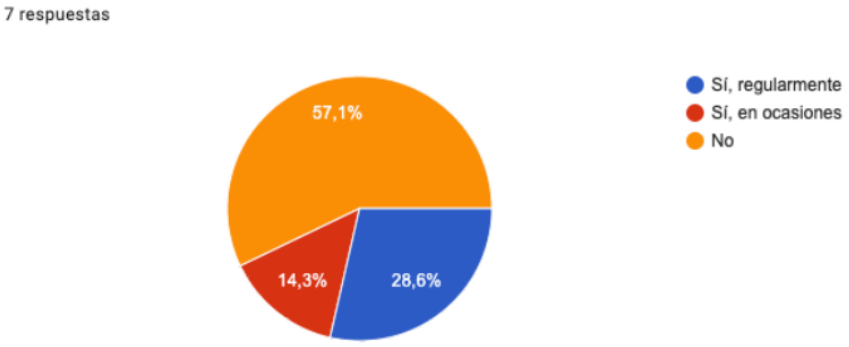


Figura B.7: ¿Ha utilizado alguna vez herramientas para simulaciones de dosis de medicamentos?

B.3. Resultados

B.3.1. ¿Qué tan útil cree que es una plataforma que combine simulaciones avanzadas y datos en tiempo real de pacientes?

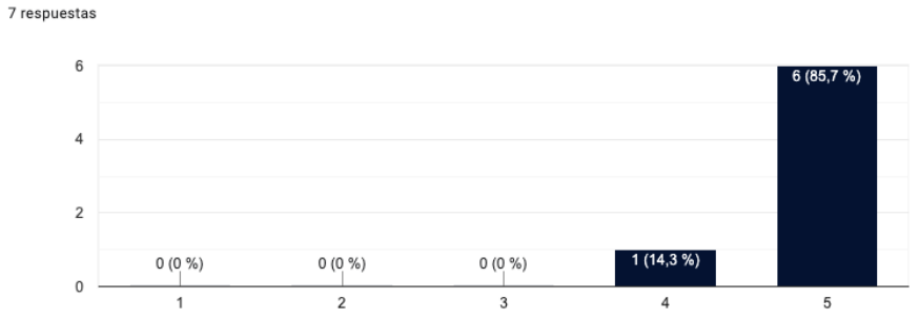


Figura B.8: ¿Qué tan útil cree que es una plataforma que combine simulaciones avanzadas y datos en tiempo real de pacientes?

B.3.2. Según su experiencia y opinión, ¿qué características considera esenciales en una herramienta de dosificación de precisión?

- No soy usuaria. Diría que es importante poder correr múltiples simulaciones variando parámetros para ver su influencia.
- Amigable y fácil de usar, poder elegir el modelo a ejecutar, poder comparar simulaciones, datos del paciente actualizados automáticamente.
- Los modelos que se utilizan deben tener validación para la población objetivo. Además, se deben considerar:
  - El almacenamiento de datos (aspectos éticos, protección de información, disponibilidad de datos para futuros análisis, herramientas de visualización).
  - Facilidad de uso y accesibilidad remota.
- Debe ser amigable y versátil.

B.3.3. ¿Qué limitaciones o problemas identifica actualmente en las herramientas existentes relacionadas con la dosificación de precisión, si es que utiliza alguna o está en conocimiento de su funcionamiento?

- No soy usuaria.
- Falta de integración con historia clínica de muchas herramientas (algunas sí lo tienen, pero no en Uruguay). Además, dificultad para integración con otros sistemas por la falta de estándares y la heterogeneidad.
- Falta resolver cómo y dónde se almacenan datos individualizados. Es crucial la validación de modelos para confiar en las predicciones, así como la divulgación y evaluación prospectiva de las herramientas.
- Las herramientas actuales suelen ser de difícil manejo para usuarios que no están acostumbrados a trabajar en dosificación de precisión. Generalmente, el usuario final debe tener un conocimiento básico para poder utilizarlas. Considerando que muchos usuarios son médicos que no trabajan frecuentemente con modelos, lograr una plataforma de sencillo manejo es fundamental.
- Hay pocas plataformas que incluyan las características mencionadas anteriormente. Además, muchas herramientas requieren manejo de programas específicos que implican conocimientos avanzados.

B.4. Resultados

B.4.1. ¿Le pareció innovadora la idea de aplicar gemelos digitales en la salud?

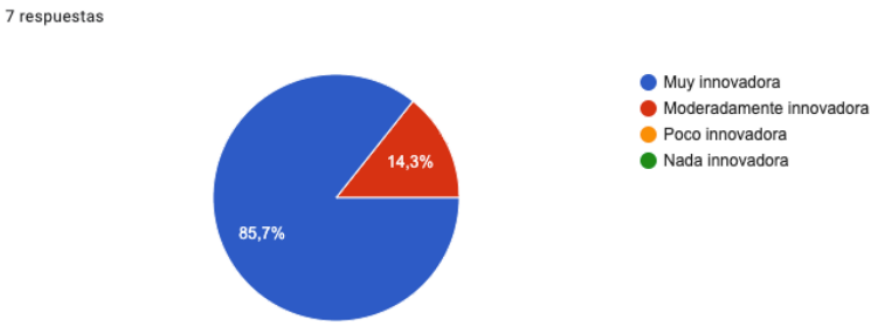


Figura B.9: ¿Le pareció innovadora la idea de aplicar gemelos digitales en la salud?

B.4.2. ¿Qué tan importante considera que fue la integración de datos en tiempo real desde dispositivos como *wearables*?

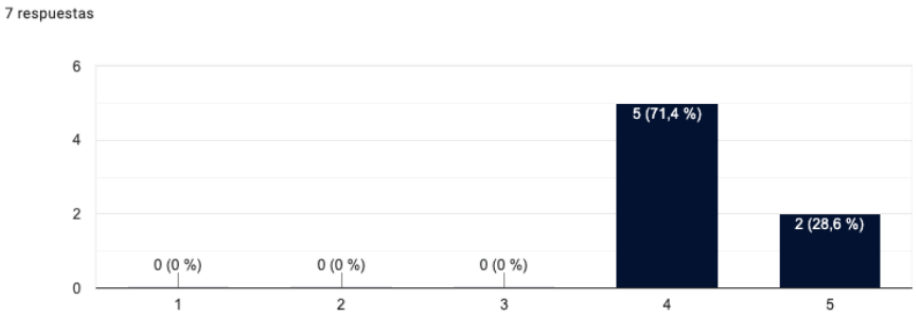


Figura B.10: ¿Qué tan importante considera que fue la integración de datos en tiempo real desde dispositivos como *wearables*?

B.4.3. ¿Qué tan relevante es para usted que una plataforma sea fácil de usar en la práctica médica diaria?

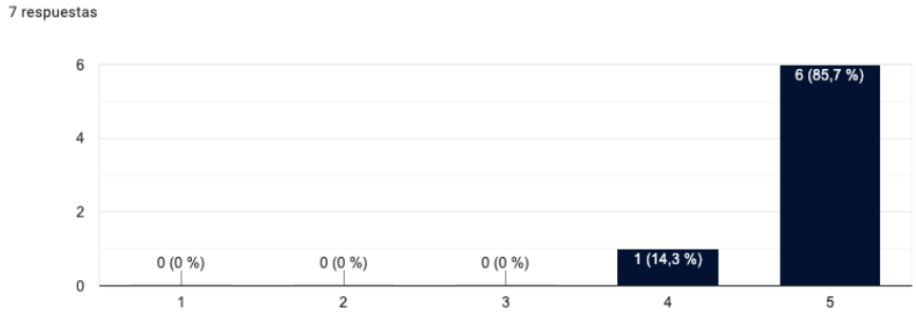


Figura B.11: ¿Qué tan relevante es para usted que una plataforma sea fácil de usar en la práctica médica diaria?

B.4.4. ¿Considera que la plataforma presentada es intuitiva y sencilla de utilizar?

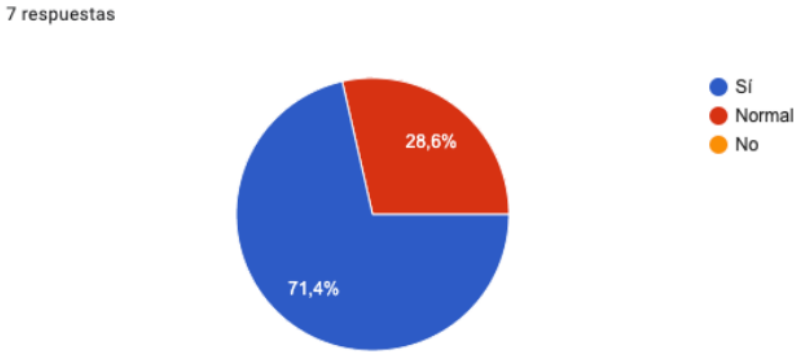


Figura B.12: ¿Considera que la plataforma presentada es intuitiva y sencilla de utilizar?

B.4.5. ¿Considera que la plataforma presentada es visualmente agradable?

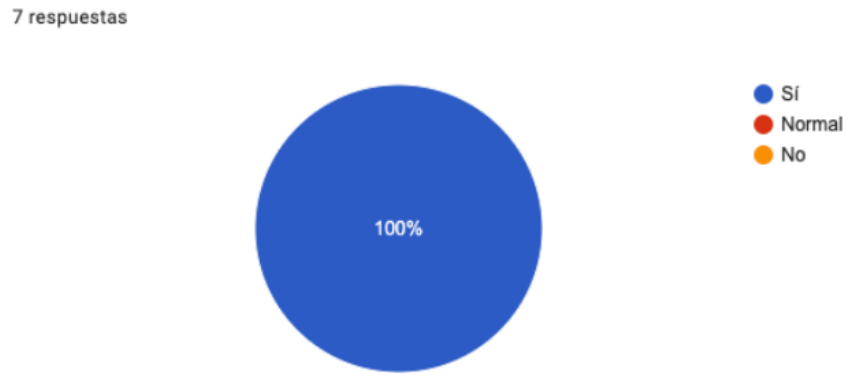


Figura B.13: ¿Considera que la plataforma presentada es visualmente agradable?

B.4.6. ¿Cuáles funcionalidades considera prioritarias en esta plataforma?

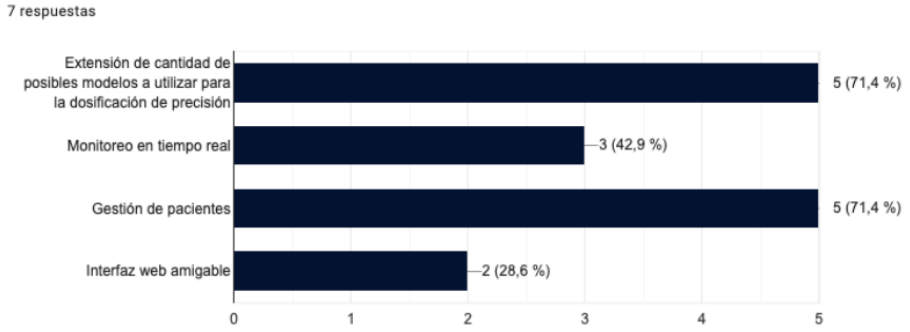


Figura B.14: ¿Cuáles funcionalidades considera prioritarias en esta plataforma?

B.5. Validaciones

B.5.1. ¿Utilizaría esta plataforma si fuera un médico o químico involucrado?

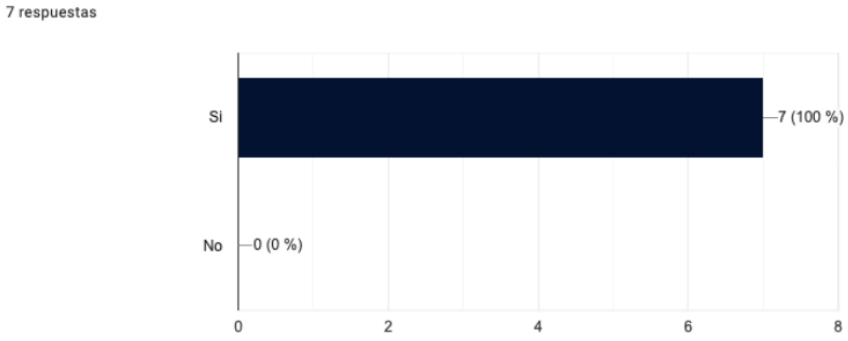


Figura B.15: ¿Utilizaría esta plataforma si fuera un médico o químico involucrado?

B.5.2. ¿Considera que es una plataforma segura?

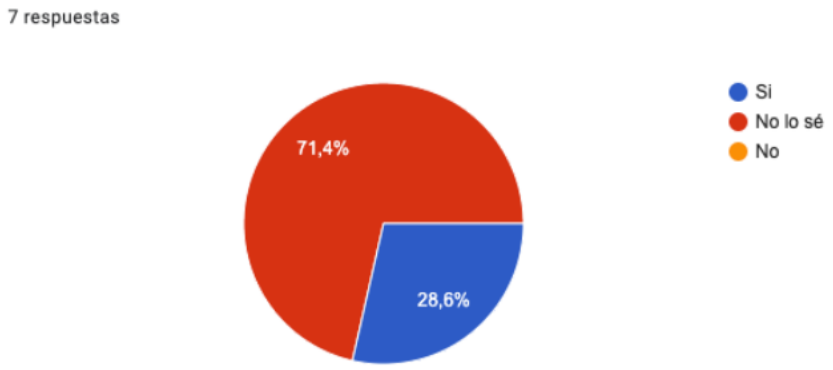


Figura B.16: ¿Considera que es una plataforma segura?

B.6. Mejoras

B.6.1. ¿Piensa que se podría haber agregado alguna otra funcionalidad más importante que las integradas? Si es así, ¿cuál?

- No.
- No vi si se puede hacer una simulación si no se tiene ningún dato cargado, o solo se tienen muy pocos datos como, por ejemplo, el género y el peso. En ese caso, creo que algunas herramientas permiten simular tomando para las variables indefinidas un valor promedio de la población.
- El uso de SNOMED, como fue mencionado, creo que es importante.
- Agregar datos del contexto del paciente relativos a comedicación y hábitos de alimentación (como el uso de hierbas medicinales, por ejemplo) que pueden explicar eventos.
- En un principio, no.

B.6.2. ¿Tiene alguna sugerencia en cuanto a la usabilidad de la plataforma?

- Tratar de poner combos de opciones en la mayor cantidad de campos posibles. Eso reduce variaciones y errores en el ingreso de datos.

- Ninguna. Parecía fácil de usar, con información concisa en cada una de las pantallas (no hay sobrecarga de información en alguna pantalla, lo cual puede hacerlo difícil de leer).
- Generar historial visual de las variables del paciente (líneas de tiempo).
- Investigar la interacción con plataformas de gestión farmacoterapéutica que existen en el mercado.
- Entiendo que puede ser útil abstraer el concepto de proveedor mediante el uso de modelos. Con esto, se seleccionan modelos y, si hay modelos exclusivos, se utiliza un proveedor determinado, pero escondiendo esa decisión. Para esto, también se debe trabajar en homologar los parámetros requeridos, para poder intercambiar fácilmente de proveedor, siempre y cuando se respete la interfaz.

B.6.3. ¿Tiene alguna otra opinión acerca de mejoras o correcciones de la plataforma?

- No, ya lo tienen en trabajos futuros. (Para demos, que no tenga el fondo de constelaciones móviles, distrae del centro de la UI).
- Ninguna. ¡Muy buen trabajo!
- Si bien ya está conectada a FINGLIX, creo que la conexión de FINGLIX con esta plataforma para realizar simulaciones individuales en tiempo real es fundamental. Asimismo, que estas simulaciones sean guardadas también en la plataforma para poder dar seguimiento a los pacientes.
- Mencionaron que agregar autenticación con usuario gub.uy ofrecería mayor seguridad. Entiendo que no sería necesariamente cierto. En particular, hay que ver qué mecanismos de seguridad se agregan a la base de datos de su desarrollo, ya que son datos personales.
- No sé si evaluaron mecanismos de impersonalización a la hora de comunicarse con un proveedor, para evitar que se propague información sensible y solamente sean datos que no sean asociables a usuarios.

Anexo C

Anexo Configuración

C.1. READMEs

C.1.1. Frontend

Prerrequisitos

Es necesario contar con el gestor de paquetes NPM para poder ejecutar el *frontend* de manera local.

Inicializar

1. Clonar el repositorio:

```
git clone git@github.com:GebelusDigitales/medfing-app
```

2. Moverse al repositorio:

```
cd medfing-app
```

3. Instalar bibliotecas:

```
npm i
```

4. Correr la versión de desarrollo:

```
npm run dev
```

C.1.2. Backend

Para inicializar el backend localmente con Express y PostgreSQL, se ejecuta el siguiente comando:

```
npm i express pg
```

Configuración de PostgreSQL

1. Instalar PostgreSQL en la computadora.
2. Crear un usuario distinto al administrador con el rol para crear bases de datos.
3. Crear una base de datos llamada `api`.
4. Instalar pgAdmin como cliente para conectar a la base de datos.
5. Conectamos el backend con la base de datos `api` mediante un archivo de configuración sincronizado al iniciar el backend. Este archivo contiene los datos necesarios para la conexión con PostgreSQL.

Creación de Modelos con Sequelize

Iniciamos con la creación del modelo `User` para los Médicos. Creamos un módulo llamado `users` y un archivo `users.model.js` definido de la siguiente forma:

```
const User = db.define(
  "user",
  {
    name: {
      type: Sequelize.STRING,
      allowNull: false,
      unique: true,
    },
    email: {
      type: Sequelize.STRING,
      allowNull: false,
      unique: true,
    },
    password: {
      type: Sequelize.STRING,
      allowNull: false,
      unique: true,
    },
    role: {
      type: Sequelize.INTEGER,
      defaultValue: 0,
    },
  },
  { paranoid: true }
);
```

Con esto, ya tenemos el modelo de usuario creado en la base de datos.

Estructura de Servicios

Para interactuar con la clase `Users`, usamos tres archivos: rutas, controlador y servicio.

En el archivo `users.routes.js` se definen las rutas HTTP para interactuar con `Users`. Ejemplo:

```
// Rutas para CRUD de usuarios
router.get('/users', usersController.getAllUsers);
router.post('/users', usersController.createUser);
router.delete('/users/:id', usersController.deleteUser);
```

En `users.controller.js` se implementan las funciones que manejan las solicitudes HTTP:

```
// Ejemplo: Borrar usuario
async function deleteUser(req, res) {
  try {
    const userId = req.params.id;
    await usersService.deleteUser(userId);
    res.status(200).send("Usuario eliminado");
  } catch (error) {
    res.status(500).send("Error al eliminar usuario");
  }
}
```

En el archivo `users.service.js` se encuentra la lógica de negocio principal:

```
// Servicio para eliminar un usuario
async function deleteUser(userId) {
  return await User.destroy({ where: { id: userId } });
}
```

Utilidades y Validaciones

En una carpeta llamada `utils`, se crean archivos para:

- Validaciones de tipos y rangos.
- Manejo de errores comunes.
- Funciones compartidas.
- Constantes utilizadas en rutas.

Autenticación y Tokens

Se crea un módulo `auth` con rutas `POST` para registro y login, utilizando `jwt` para tokens y `md5` para encriptar contraseñas.

Pruebas

Probamos los módulos creados con herramientas como `Postman` para verificar las operaciones del backend.

Para ejecutar las pruebas unitarias y visualizarlas en una interfaz gráfica se debe ejecutar el comando:

```
    npx vitest --ui --coverage
```

C.1.3. R y Mgrsolve

Instalar R

- Instalar R desde CRAN.
- Instalar Rtools (solo necesario si se usa Windows).

Instalar los Paquetes Requeridos

Primero, asegurarse de tener los paquetes necesarios instalados:

```
install.packages("mrgsolve")
install.packages("dplyr")
install.packages("ggplot2")
```

Crear modelos

Crear un archivo `.cpp` para el modelo, que define los compartimentos, parámetros y ecuaciones diferenciales. Este es un ejemplo básico de un modelo PBPK humano:

```
$PROB
$PARAM
  BW = 70,  // Peso corporal (kg)
  Qliver = 1.5,  // Flujo sanguineo hepatico (L/h)
  Vblood = 5,  // Volumen sanguineo (L)
  Cliver = 0.5  // Depuracion hepatica (L/h)
$CMT
  BLOOD LIVER
$ODE
  dxdt_BLOOD = -Qliver * BLOOD / Vblood + Qliver * LIVER / Vblood;
  dxdt_LIVER = Qliver * BLOOD / Vblood - (Qliver + Cliver) * LIVER
      / Vblood;
$TABLE
  double CP = BLOOD / Vblood;
$CAPTURE CP
```

Guardar este modelo como `pbpk_model.cpp`.

Luego, usar `mrgsolve` para cargar y compilar el modelo:

```
library(mrgsolve)
library(dplyr)
library(ggplot2)

mod <- mread("pbpk_model", file = "path/to/pbpk_model.cpp")
```

Configurar las condiciones iniciales y los parámetros para la simulación:

```
# Condiciones iniciales
init <- c(BLOOD = 0, LIVER = 0)

# Tiempo de simulacion
times <- seq(0, 24, by = 0.1)  # simula durante 24 horas

# Informacion de la dosificacion
```

```
dose <- ev(amt = 100, cmt = 1, time = 0)  # dosis de 100 mg al
    compartimento BLOOD

# Ejecutar la simulacion
out <- mod %>% ev(dose) %>% init(init) %>% mrgsim(end = 24, delta =
     0.1)
```

Graficar los resultados de la simulación para visualizar la evolución concentración-tiempo:

```
# Convertir salida a un data frame
out_df <- as.data.frame(out)

# Graficar los resultados
ggplot(out_df, aes(x = time, y = CP)) +
  geom_line() +
  labs(title = "Perfil Concentracion-Tiempo", x = "Tiempo (horas)",
       y = "Concentracion en Plasma (mg/L)")
```

Para modelos más complejos, se pueden agregar más compartimentos y procesos como el metabolismo, la excreción y diferentes rutas de administración, para lo que sería necesario actualizar el archivo `.cpp`.

C.2. Eclipse Ditto

Guía para ejecutar Eclipse Ditto localmente. Es necesario contar con una versión instalada de Docker para poder inicializar Eclipse Ditto de manera local.

C.2.1. Inicializar

1. Clonar el repositorio:

```
git clone https://github.com/eclipse-ditto/ditto.git
```

2. Moverse al repositorio:

```
    cd eclipse-ditto
```

3. Construir los containers de Docker:

```
    docker compose build
```

4. Iniciar los containers:

```
    docker compose up
```

5. Ejecutar Ditto:

```
    docker compose run eclipse-ditto
```

6. Crear una terminal de bash en Ditto:

```
    docker compose run eclipse-ditto bash
```

Si la build falla, vale la pena borrar las imágenes, volúmenes y containers de Docker:

```
    docker system prune -a
```

Esto va a borrar todos los recursos que están "flotando"(no asociados a un container corriendo), además de que con la tag `-a` se le agregan todos los containers que no están corriendo y las imágenes que no se están usando.

C.2.2. Pruebas de concepto

Prerrequisitos

Es necesario obtener las dependencias necesarias para instalar Mosquitto y para contar con el simulador de reloj inteligente.

- *Pull* de Mosquitto:

```
docker pull eclipse-mosquitto
```

- Clonar el repositorio con el simulador de iWatch:

```
git clone https://github.com/bernar0507/Eclipse-Ditto-MQTT-
    iWatch.git
```

- Es una buena práctica correr:

```
docker system prune -a
```

para limpiar todo y evitar interferencias.

Iniciar Ditto y Mosquitto

Para iniciar Ditto, ir al directorio `deployment/docker` y ejecutar:

```
docker-compose up -d
```

Configuración de Mosquitto

Editar el archivo `mosquitto.conf` agregando:

```
allow_annonymous true
```

Iniciar Mosquitto con:

```
docker run -it --name mosquitto -p 1883:1883 -v $(pwd)/mosquitto:/
    mosquitto/ eclipse-mosquitto mosquitto -c ./mosquitto/config/
    mosquitto.conf
```

Crear la policy

Ejecutar este comando para definir las reglas:

```
curl -X PUT 'http://localhost:8080/api/2/policies/android:policy' -u 'ditto:ditto' \
-H 'Content-Type: application/json' \
-d '{
    "entries": {
        "owner": {
            "subjects": {
                "nginx:ditto": {"type": "nginx basic auth user"}
            },
            "resources": {
                "thing:/": {"grant": ["READ","WRITE"], "revoke": []},
                "policy:/": {"grant": ["READ","WRITE"], "revoke": []},
                "message:/": {"grant": ["READ","WRITE"], "revoke": []}
            }
        }
    }
}'
```

Crear la conexión MQTT

Obtener la IP del container de Mosquitto:

```
mosquitto_ip=$(docker inspect -f '{{range .NetworkSettings.Networks}}{{.IPAddress}}{{end}}' mosquitto)
```

Crear la conexión con:

```
curl -X POST \
 'http://localhost:8080/devops/piggyback/connectivity?timeout=10' \
 -H 'Content-Type: application/json' \
 -u 'devops:foobar' \
 -d '{
   "targetActorSelection": "/system/sharding/connection",
   "piggybackCommand": {
       "type": "connectivity.commands:createConnection",
       "connection": {
           "id": "mqtt-connection-iwatch",
           "connectionType": "mqtt",
           "uri": "tcp://ditto:ditto@$mosquitto_ip:1883"
       }
   }
}'
```

Enviar datos desde el simulador

Instalar las dependencias:

```
pip install numpy scipy paho-mqtt
```

Correr el script:

```
python3 send_data_iwatch.py
```

Verificar la conexión:

```
curl -u ditto:ditto -X GET 'http://localhost:8080/api/2/things/org.
    Iotp2c:iwatch'
```

Con esto, Ditto y Mosquitto están configurados.

C.3. Despliegues

C.3.1. Despliegue del backend

Para desplegar el backend se debe crear un entorno de tipo *Docker Engine* en Elastic cloud. La mayor versión de Docker que Elastic Cloud nos permite es la 19, por lo que al crear el entorno se debe seleccionar esa.

Clonar el repositorio del backend en la máquina de Elastic Cloud:

```
git clone git@github.com:GebelusDigitales/medfing-node-api.git
```

Actualizar la versión de Docker de la máquina usando el gestor de paquetes `yum`:

```
sudo yum install docker-ce docker-ce-cli containerd.io docker-
    buildx-plugin docker-compose-plugin
```

Construir la imagen usando `docker-compose`:

```
docker-compose build
```

Ejecutar la imagen:

```
docker-compose up
```

C.3.2. Despliegue de Ditto

Dado que ejecutar Eclipse Ditto requiere un uso considerable de recursos en la máquina *host*, se decide utilizar un código promocional de Mi Nube para desplegar un servidor de Ditto en Elastic Cloud. La estrategia inicial consistía en realizar el despliegue mediante `docker-compose`[1], pero se encontraron diversas dificultades con este enfoque. A pesar de asignar más recursos a la máquina en la nube, los contenedores no lograban iniciar. Tras revisar los registros del sistema, se identifica que el problema se debía a la falta de memoria. El código promocional de Mi Nube equivale a un uso mensual de cinco mil pesos uruguayos, por lo que la memoria asignable no es ilimitada, y Eclipse Ditto tiene requerimientos bastante altos con respecto a recursos, tales como dos núcleos de CPU disponibles y cuatro gigabytes de memoria dedicados. (T. E. Foundation, 2024)

Investigando más a fondo, se descubre que los desarrolladores de Ditto no recomiendan el uso de docker-compose para entornos de producción, ya que esta solución fue creada para un entorno local. Esto lleva a explorar otras alternativas, como la utilización de Kubernetes.

[1] https://docs.docker.com/compose/

Conceptos relacionados con Kubernetes

Kubernetes es una plataforma de código abierto diseñada para automatizar la implementación, escalado y operación de aplicaciones en contenedores. Permite gestionar aplicaciones en microservicios mediante herramientas que facilitan la orquestación de contenedores, abarcando la administración de su ciclo de vida, la disponibilidad, la escalabilidad y la recuperación ante fallos (T. L. Foundation, 2024).

En el contexto de Kubernetes, el **Control Plane** es el componente encargado de dirigir el clúster de Kubernetes. Administra el estado deseado del sistema y coordina el funcionamiento de los nodos en el clúster, asegurando que todo funcione según las especificaciones definidas.

Los **Nodos** son las máquinas que ejecutan las aplicaciones en contenedores y otros componentes del clúster. Un clúster de Kubernetes está compuesto por varios nodos que trabajan de forma conjunta para ejecutar estas aplicaciones.

Los **Contenedores** son unidades de software que empaquetan el código de una aplicación junto con todas sus dependencias y configuraciones necesarias para su ejecución, de manera similar a los contenedores de Docker.

Los **Pods** son la unidad básica de ejecución en Kubernetes. Un *Pod* puede contener uno o más contenedores que comparten el mismo espacio de red y almacenamiento. Son efímeros, lo que significa que Kubernetes los reemplaza y administra automáticamente según sea necesario para mantener la aplicación operativa.

Helm es una herramienta que facilita la instalación y gestión de aplicaciones en Kubernetes mediante charts. Helm Chart es un paquete preconfigurado de Kubernetes que incluye todo lo necesario para desplegar una aplicación, como los archivos de configuración. Facilita la gestión de aplicaciones complejas y su despliegue en diferentes entornos.

kubectl es la herramienta de línea de comandos para interactuar con Kubernetes, permitiendo a los usuarios desplegar y gestionar aplicaciones e inspeccionar recursos.

Finalmente, el **ingress** es un recurso de Kubernetes que gestiona el acceso externo a los servicios en un clúster, generalmente mediante HTTP o HTTPS. Actúa como punto de entrada para las aplicaciones ejecutadas en el clúster.

Despliegue mediante Kubernetes

El despliegue de Eclipse Ditto en Kubernetes se realiza a través de una serie de pasos. Primero, se añade el repositorio de Helm correspondiente a la aplicación de Ditto publicado por sus desarrolladores y se actualiza para asegurarse que se tiene la versión más reciente.

Una vez hecho esto, se configuran los recursos de memoria. Es necesario ajustar el Helm Chart para limitar la memoria disponible para cada *pod*, ya que una asignación excesiva puede causar problemas similares a los enfrentados con Docker, donde los contenedores no podían iniciar debido a la falta de recursos. Para solucionar esto, se reduce la memoria máxima asignada a cada

pod a 512 *megabytes*, lo que resulta suficiente para mantener la aplicación en funcionamiento sin consumir recursos de forma excesiva. También se ajusta el Helm Chart para modificar otras configuraciones, tales como la configuración de un usuario con contraseña cifrada mediante *hashing* para permitir el acceso a los datos y otro de administrador para modificar configuraciones mediante el protocolo HTTP.

Posteriormente, para permitir el acceso externo a Eclipse Ditto a través de la web, se configura un recurso de *ingress*. Esto actúa como un punto de entrada, permitiendo acceder a la aplicación desde el navegador y comprobar que el despliegue se ha realizado de forma correcta y que la aplicación está operativa.

Finalmente, se despliega otro servidor en el que corre Mosquitto, el *broker* que se encarga de manejar las conexiones MQTT. Se crea una relación entre el *broker* y Ditto mediante configuraciones HTTP usando el usuario de administrador que fue configurado en la Helm Chart.

Para desplegar Ditto, seguimos los siguientes pasos:

1. Agregamos el repositorio de Helm correspondiente:

```
helm repo add eclipse-iot https://eclipse.org/packages/
    charts
```

2. Actualizamos el repositorio:

```
helm repo update
```

3. Se modifica el Helm chart para reducir la memoria de todos los pods a un máximo de 512 MB, evitando problemas relacionados con recursos. Para ello, se obtiene el archivo `values.yaml` del repositorio oficial de Ditto y lo usamos con este comando:

```
helm install eclipse-ditto eclipse-iot/ditto -f values.
    yaml
```

4. Configuramos el `Ingress` para permitir el acceso web, agregando nuestro pod de nginx:

```
kubectl create ingress ditto-ingress --rule=pg-dt-kube.web
    .elasticloud.uy/*=eclipse-ditto-nginx:8080
```

Esto nos permitió acceder a Ditto desde la web y comprobar que quedó correctamente configurado.

Comandos útiles

- Para obtener todos los pods, nodos o información general:

```
kubectl get [pods|nodes|all]
```

- Para conectarse a un pod específico:

```
kubectl exec --stdin --tty --container wait-for-gateway
    pod/eclipse-ditto-gateway-5d964f445d-h6cn8 -- /bin/sh
```

- Para controlar el `Ingress`:

```
kubectl get ingress
```

- Para obtener los logs de un pod:

```
kubectl logs <nombre_pod>
```

- Para desinstalar Ditto (por ejemplo, para actualizar los valores):

```
helm uninstall eclipse-ditto
```